ЗМІСТ

ВСТУП

Сучасна структура корпоративних мереж все більш ускладнюється, як у напрямі моніторингу трафіку, так і у напрямі використання різного устаткування. Тому системному адміністратору важко прийняти рішення, так як він повинен вивчити велику кількість даних про різні факти, які бувають не пов'язані один з одним, за короткий проміжок часу. Це призвело до появи мультиагентних систем.

Мультиагентна система допомагає приймати рішення при роботі з завданнями, які не мають чіткої структури або мають кілька критеріїв. Для створення мультиагентної системи потрібно вивчити багато дисциплін, таких як теорії штучного інтелекту, бази даних, методи імітаційного моделювання тощо.

На сьогодняшній день розробка структури мультиагентої системи сильно залежить від різновиду завдання, для якого вона створюється, а також від інформації про неї і користувачів, для яких ця система розробляється.

Мультиагента система - це система, утворена декількома взаємодіючими інтелектуальними агентами. Мультиагентні системи можуть бути використані для вирішення таких проблем, які складно або неможливо вирішити за допомогою одного агента або монолітної системи.

Актуальність створення мультиагентної системи підтверджується тим, що вартість і відповідальність прийняття рішень при роботі в корпоративній мережі весь час зростає, а час на прийняття цих рішень постійно зменшується. Тому особливо корисне застосування мультиагентих систем при керуванні небезпечними виробничими об'єктами з метою поліпшення і автоматизування процесів.

Про це свідчить масштаб і кількість техногенних аварій за минуле десятиліття, а також тяжкість наслідків після них. Крім цього, важливо враховувати інформацію про погіршення надійності та старіння обладнання, яке використовуються на виробництві. Так само, треба враховувати людський фактор

при роботі зі складними системами, так як близько 70% аварій відбуваються з вини особи, яка приймає рішення.

В даний час розвиваються нові напрямки в інформаційних технологіях. Поява інтелектуальних систем, розподільних технологій і багато чого іншого змушують підприємства вдосконалюватися.

Можна стверджувати, що поліпшити ефективність мультиагентної системи при риботі в корпоративній мережі можна за допомогою переходу до систем накопичення знань і орієнтування на прогнозування.

Складність розробки мультиагентної системи для корпоративної мережі визначена тим, що з'являється необхідність створення складних баз знань і баз даних.

Виходячи з цього, виникає необхідність розробити таку мультиагенту систему, яка буде допомагати та полегшувати роботу особі, яка приймає рішення, а саме – системному адміністраторові, при роботі з великими об’ємами інформації у корпоративній мережі.

Мета роботи: Допомога та полегшення умов роботи для особи, яка приймає рішення при роботі в корпоративній мережі за допомогою розробки мультиагентної системи.

Завдання, які вирішуються в роботі:

– Аналіз існуючих мультиагентних систем для полегшення роботи з великими об’ємами інформації в корпоративній мережі.
– Розробка мультиагентної системи.
– Аналіз мультиагентної системи.

В розділі 1 розглядаються основні види мультиагентих систем. Проаналізовано поняття «агент», «інтелектуальний агент» та «мультиагента система». Визначено, що агентом є обчислювальна система, вміщена в зовнішнє середовище, здатна взаємодіяти з нею, роблячи автономні раціональні дії для

досягнення цілей. Визначено, що навідміну від звичайного агента інтелектуальний агент повинен мати властивості реактивності проактивності та соціальності.

Введено визначення мультиагентної системи, яка представляється кінцевою множиною станів зовнішнього середовищ; кінцевою множиною агентів, кожен з котрих представлений розширеною моделлю інтелектуального агента; функцією, що описує можливу реакцію зовнішнього середовища на дії всіх агентів системи.

Наведено та проаналізовано основні методи та різновиди мультиагентих систем:

- метод класної дошки і його варіації;
- метод концепції істот, розроблений Дугласом Ленатом;
- метод концепції акторів;
- повноцінна мультиагентна інтелектуальна система, описана Гессером.

Розглянуто, для яких основніх типів систем використовуються мультіагентіе системе:

- Відкриті системи;
- Складні розподілені системи;
- Інтерактивні системи.

Відзначено, що в промисловості мультиагентні системи найбільш поширені в автоматизації управління складними системами, зборі та обробці інформації, іграх.

Наведено підходи до розробки мультиагентних систем, що відрізняються в залежності від класу задач, для вирішення яких ця система призначена, а самє:

- Gaia;
- SODA;
- Tropos.

Поставлено мету розробити мультиагентую систему підтримки прийняття рішень для работи в корпоративній мережі, з метою допомагати адміністратору приймати рішення при роботі з великими об’ємами інформації.

В розділі 2, опираючись на наукове дослідження Д.А. Белозерова, розглянуто використання агентного підходу для побудови мультиагентної системи при роботі з обчислювальною мережею. Розглянуто функціональні вимоги до мультиагентої системи з управління обчислювальною мережею. Виділено модель FCAPS, яка широко використовується при розробці систем управління мережами зв'язку та телекомунікаціями. Проведено огляд основних агентів, які необхідно використовувати для розробки мультиагентної системи підтримки прийняття рішень. Також, проведено огляд архітектури та мови реалізації системи. Наведено перелік особливостей використання агентної платформи JADE при розробці мультиагентої системи. Розглянуто онтологію мультиагентої системи при роботі в корпоративній мережі.

В якості експерименту розглянуто модель мультиагентної системи підтримки дистанційного навчання, яка має у своїй взаємодії трьох осіб – слухача, викладача та адміністратора.

За основу взято систему IDEAL (The intelligent distributed environment for active learning) [3]. Система підтримує персоніфіковану взаємодію користувачів з навчальною системою, допомагає в автоматичній оцінці результатів, і забезпечує прості у використанні засоби розробки інтерактивних мультимедійних додатків. Наведено прототип реалізацій моделі агентної КСПДО.

Показано наступні діагарми:

- діаграма фізичної структури системи;
- діаграма взаємодії агентів навчання;
- діаграма взаємодії класів агента навчання з контролером, диспетчером і сервером агентів;
- діаграма взаємодії класів агента повідомлень, агента навчання, Mail-агента і агента пошуку;
- діаграма обміну повідомленнями між агентами пошуку, навчання, повідомлень, mail;
- діаграма обміну повідомленнями між сервером мобільних агентів і

агентами, а також контролером, диспетчером і файл-сервером.

В якості висновку отримано такі нові наукові результати: досліджено систему дистанційного навчання як модель мультиагентної системи; розроблено модель дистанційного навчання на базі агентної архітектури JADE (Java Agent Development Framework); побудовано модель дистанційного навчання в середовищі проектування Rational Rose Professional Edition на мові UML (Unified Modeling Language); згенеровано каркас програмного коду системи за мовою програмування Java.

В розділі 3 запропоновано розробити модель мультиагентної системи підтримки прийняття рішень в корпоративній мережі. На підставі аналізу виділено наступні функції системи.

• Виявлення пристроїв.

• Моніторинг мережі.

• Зберігання даних моніторингу.

• Відображення даних моніторингу.

• Управління відмовами.

• Повідомлення про події.

Для вирішення перерахованих завдань виділено кілька типів агентів:

- агент інтерфейсу;
- агент адміністрування (управління);
- агент формування звітів;
- агент зберігання даних;
- агент моніторингу;

Наведено базову архітектуру мультиагентної СППР. Описано структурну модель системи. Наведено концептуальну модель системи. Розроблено базу правил для реалізації мультиагентної СППР. Основним засобом розробки обрано мережі Петрі - математичний апарат, який допомагає моделювати динамічні дискретні системи. Враховано підходи, такі як: прийняття рішень, вилучення даних, відображення даних, побудова системи. При проектуванні використано

мультиагентну СППР на основі нечіткого виведення. В ході роботи прийнято рішення поліпшити схему системи шляхом прискорення передачі сигналу від особи, яка приймає рішення до агенту моніторингу. Вдосконалено базу правил(БП) для покращення її роботи.

Проведено аналіз мультиагентної системи підтримки прийняття рішень.При дослідженні отриманої системи було виявлено, що при виправленні схеми, коли ОПР безпосередньо передає дані до ЛОМ, система працює швидше і точніше. При впровадженні в систему генераторів і термінаторів з'явилася можливість подавати нерівномірно надходячі сигнали в необмеженій кількості.

Створення мультиагентної СППР дозволяє вирішити проблему, яка пов'язана з людським фактором і надає ОПР рішення, на основі якого користувач витрачає менше часу на аналіз існуючої інформації. У даній роботі було розроблено та показано, як можна удосконалити мультиагентну СППР з урахуванням взаємодії різних компонентів в мережі. Дана модель мультиагентої СППР здатна давати швидке рішення користувачу на основі повної та оперативної інформації і давати звіт про стан системи в цілому.

У розділі 4 проведено аналіз умов праці на робочому місці програміста. Перевірено, чи відповідають параметри мікроклімату, іонного складу повітря, змісту шкідливих речовин на робочих місцях, вимогам ДСН 3.3.6-042-99 «Державним санітарним нормам мікроклімату виробничих приміщень». Проведено аналіз шуму в робочій зоні та освітлення робочої зони. Розроблено заходи з охорони праці, а саме:

- організаційні заходи;
- медико-профілактичні заходи;
- заходи щодо забезпечення чистоти повітря робочої зони.

Розглянуто інженерні (технічні) рішення з охорони праці. Розраховано штучну загально-обмінну вентиляцію.

Описано пожежну безпеку приміщення.

В якості індивідуального завдання проведено:

- розрахунок максимального значення надлишкового тиску, що очікується в районі цеху;
- розрахунок границі стійкості цеху до дії ударної хвилі;
- визначення можливих наслідків вибуху – ступені руйнувань елементів цеху.

1 АНАЛІТИЧНИЙ ОГЛЯД МУЛЬТИАГЕНТНИХ СИСТЕМ

1.1 Аналіз поняття агенту та мультиагентної системи.

В основі будь-якої мультиагентої системи лежить поняття «агент». Це поняття останнім часом було адаптовано до багатьох областей як прикладного і системного програмування, так і до досліджень в областях штучного інтелекту і розподілених інтелектуальних систем. Причому в кожному конкретному випадку поняттю надавалося різне значення. Необхідно зафіксувати поняття "агента", яке буде використовуватися далі в роботі. Основною відмінною властивістю агентів є те, що агенти здійснюють дії. Далі часто стверджується, що агенти не просто здійснюють дії, але вони діють автономно і раціонально. Під автономністю зазвичай розуміють, що агент діє без прямого втручання людини або іншої керуючої суті.

З поняттям раціональності все не так просто, але часто під раціональністю розуміють прагнення агента оптимізувати значення деякої оціночної функції. Міра раціональності неявно вказує на те, що агент має мети, яких "хоче" досягти, і уявлення про навколишній світ, на котрі агент спирається при виборі дії.

Ще однією важливою властивістю агента є те, що він поміщений під зовнішню середу, з якою він здатний взаємодіяти. Як правило, середовище не контролюється агентом, тобто він здатний лише впливати на нього, але не контролювали повністю.

У підсумку, можна сформувати таке визначення агента, адаптоване багатьма сучасними дослідниками:

Агент - обчислювальна система, вміщена в зовнішнє середовище, здатна взаємодіяти з нею, роблячи автономні раціональні дії для досягнення цілей.

Очевидно, що перераховані вище властивості не є достатніми для визначення інтелектуального агента, так як вони не передбачають явно гнучкості поведінки.

Зазвичай для того, щоб вважатися "інтелектуальним" агент повинен володіти такими властивостями:

– Реактивність (reactivity) - агент повинен відчувати зовнішнє середовище і реагувати на зміни в ньому, здійснюючи дії, спрямовані на досягнення цілей.

– Проактивність (pro-activeness) - агент повинен показувати керовану цілями поведінку, проявляючи ініціативу, здійснюючи дії, спрямовані на досягнення цілей.

– Соціальність (social ability) - агент повинен взаємодіяти з іншими сутностями зовнішнього середовища (іншими агентами, людьми і т.д.) для досягнення цілей.

Будь-яка з перших двох властивостей може бути досягнена досить легко. Наприклад, властивістю проактивності, в широкому сенсі, володіє будь-який компілятор, основна мета якого, очевидно, сформувати низькорівневий об'єктний код на основі програмного коду на мові програмування високого рівня. По суті, будь-яка функція може бути розглянута як система, метою якої є перетворення деяких вхідних даних у вихідні. Однак, очевидно, що зміна у вхідних даних функції під час її роботи практично напевно призведе до краху або, як мінімум, невідповідності вихідних даних вхідним. Тобто, ні функції, ні компілятори (які, безумовно, можна розглядати як різновид функцій) не володіють властивістю реактивності.

З іншого боку, тільки реактивні системи (тобто системи які тільки реагують на зміни у зовнішньому середовищі) теж досить прості – навіть найпримітивніша продукційна система демонструє реактивність. Совміщати в системі обидві властивості в потрібних пропорціях є непростим завданням. Буде не дуже ефективно, якщо агент жорстко слідує сценарію досягнення мети, не реагуючи на зміни у зовнішньому середовищі і не володіючи можливістю помітити необхідність коригування плану. Але так само не ефективною буде і поведінка, обмежена лише

реакцією на надходячі ззовні стимули, без будь-якого планування цілеспрямованих дій.

Насправді описана проблема настільки складна, що навіть далеко не всі люди здатні ефективно її вирішувати. Дуже часто можна побачити людину, яка кидається на кожну можливість, але ніколи не концентрується на цій можливості впродовж достатнього часу, щоб повноцінно її реалізувати. Але так само часто зустрічаються люди, які, одного разу поставив мету і сформувавши план, будуть намагатися чітко його дотримуватися, не помічаючи змін в ситуації, що вимагають перегляду планів або цілей.

Що стосується третьої властивості, соціальності, то вона теж не так проста, як може здатися на перший погляд. З одного боку, кожен день мільйони комп'ютерів взаємодіють між собою, обмінюючись пакетами бінарних даних. Але таку поведінку ще не можна вважати соціальним. Крім комунікації, соціальна поведінка має включати кооперацію з іншими сутностями, яка полягає в поділі цілей між окремими сутностями, спільне планування та координацію дій, спрямованих на досягнення загальних цілей. Соціальна поведінка як мінімум передбачає наявність у агента уявлень про цілі інших сутностей і тому, як вони планують цих цілей досягти.

Мультиагентна система представляється трійкою MAS = (S, AG, env), де:

- S є кінцева множина станів зовнішнього середовища;
- AG = $\{ag_1, \ldots, ag_n\}$ є кінцева множина агентів, кожен з котрих представлений розширеною моделлю інтелектуального агента;
- env : $S \times A_{ag1} \times \ldots \times A_{agn} \rightarrow 2^S$ є функція, що описує можливу реакцію зовнішнього середовища на дії всіх агентів системи. Множину всіх можливих спільних дій системи позначимо $ACS = A_{ag1} \times \ldots \times A_{agn}$.

Розширена модель агента відображає наступні додаткові властивості мультиагентної системи:

Комунікація агентів. Можна виділити два основних види комунікації: оперативна комунікація, яка використовується агентами для координації своїх дій

в поточний момент, і Високорівнева комунікація, яка використовується агентами для обміну інформацією.

Оперативна комунікація. Так як в рамках оперативної комунікації агентам потрібно максимально швидко обмінятися відносно невеликою кількістю інформації, для моделювання цього виду комунікації найкраще підійде модель сигналів.

Для кожного з агентів ag_i системи визначено кінцеву множину сигналів Sig_{agi}, а також функцію $Send_{agi} : P_{agi} \times AG \rightarrow Sig_{agi}$, що описує, які сигнали agi відправить кожному з інших агентів в даній ситуації. Кожен з агентів може відправити кожному іншому не більше, ніж одни сигнал. Для моделювання ситуації, коли агент не посилає сигнал іншому агентові, введено виділений фіктивний сигнал $sg^{ø} \in Sig_{agi}$.

Поведінка функції $Send_{agi}$ змінюється в процесі взаємодії із зовнішнім середовищем, а послані і отримані сигнали впливають на прийняття іншими агентами рішення про дію. Отже, функцію Sendagi зручно моделювати в контексті поточного плану агента. В цьому випадку план агента представляється як взаємодія кінцевого автомата $plan_{agi} = (P_{agi} ,A_{agi} , Sig_{ag1} \times . . . \times Sigagn, Sig_{agi} , I_{pln}, send_{pln}, \sigma_{pln}, i_{pln},0)$, де:

- P_{agi} є вхідний алфавіт автомата, що співпадає з безліччю можли
- них сприйняття агентом agi станів зовнішнього середовища;
- A_{agi} є вихідний алфавіт автомата, що співпадає з безліччю дій агента ag_i;
- $Sig_{ag1} \times . . . \times Sig_{agn}$ є множина вхідних сигналів автомата, що збігається з декартових добутком множин сигналів всіх агентів системи;
- Sig_{agi} є множина вихідних сигналів автомата, що збігається з множиною сигналів агента ag_i;
- I_{pln} є множина внутрішніх станів автомата;
- $send_{pln} : P_{agi} \times I_{pln} \times AG$! Sig_{agi} є функція посилки сигналів, що визначає по поточним сприйняття зовнішнього середовища і станом автомата сигнал, який буде посланий кожному з агентів системи;

– $\sigma_{pln} \subseteq P_{agi} \times Sig_{ag1} \times \ldots \times Sig_{agn} \times I_{pln} \times I_{pln} \times A$ є відношення переходів, що визначає по сприйняттю p поточного стану зовнішнього середовища, набору отриманих сигналів $(sg_1, \ldots, sg_n) \in Sig_{ag1} \times \ldots \times Sig_{agn}$ та поточного внутрішньому стану плану i_{pln} наступний стан плану i'_{pln} та дія, яку необхідно виконати агенту;

– $i_{pln},0 \in I_{pln}$ є початковий стан автомату.

Таким чином, рішення про вибір дії і зміні стану автомата відбувається в два етапи:

– Визначення та розсилка сигналів - на цьому етапі кожен з агентів системи визначає за своїм сприйняттям поточного стану зовнішнього середовища і поточного стану свого плану, які сигнали він повинен послати іншим агентам і розсилає їх.

– Вибір дії - на цьому етапі, грунтуючись на сприйнятті поточного стану зовнішнього середовища, поточний стан плану й отриманим від інших агентів сигналах, агент вибирає свою дію і наступний стан для свого плану.

Запропонована модель комунікації досить проста в реалізації і дозволяє забезпечити швидке прийняття кожним з агентів системи рішення про поточний дії. Зауважимо, що в запропонованій моделі агент може відправити сигнал сам собі, в чому, як правило, не виникає необхідності. Ця можливість збережена з метою спрощення опису.

Високорівнева комунікація. Модель сигналів добре підходить для організації оперативної комунікації, але вона погано застосовна для організації комунікації високорівневою, так як дозволяє обмінюватися тільки відносно простий інформацією. У більшості існуючих робіт для моделювання комунікації використовується теорія мовної дії [1, 2, 3], яка розглядає комунікацію агентів як різновид їх дій. Аналогічний підхід можна застосувати і для моделювання високорівневою комунікації в нашому випадку.

Визначимо для кожного з агентів системи ставлення сприйняття дій $see_{ACS} \subseteq S \times ACS \times ACS$, визначальне доступну агенту інформацію про скоєних системою діях. Сприйняття дій є коректним якщо для будь-якого $s \in S$ відношення $see_{ACS}(s) = \{(acs, acs') \in ACS \times ACS \mid (s, acs, acs') \in see_{ACS}\}$ є відношенням еквівалентності. Далі будемо розглядати тільки коректні сприйняття.

Відношення сприйняття $see \subseteq S \times S$ і сприйняття дії $see_{ACS} \subseteq S \times ACS \times ACS$ спільно задають відношення еквівалентності seeF на множині $S \times ACS$ наступним чином:

see_F , $\{(s, acs, s', acs') \in S \times ACS \times S \times ACS \mid s' \in see(s) \text{ and } \exists s'' \in see(s) : (acs, acs') \in see_{ACS}(s'')\}$ (1.1)

При цьому відношення еквівалентності see_F задає множину класів еквівалентності PF на множині $S \times ACS$, тобто, його можно приймати як функцію $see_F : S \times ACS \rightarrow P_F$. Функцію seeF – функція повного сприйняття, а множина P_F — множина повних сприйнять. Крім того, розширимо функцію оновлення внутрішнього стану агента agi, включивши в набір параметрів загальне дія, вчинена системою:

$refine_{agi} : I \times S \times ACS \cup [\{\emptyset\} \rightarrow I$. У тому випадку, коли система тільки починає взаємодіяти із зовнішнім середовищем, замість загального дії передається ∅, так як агенти ще не робили ніяких дій.

Подальша деталізація та уточнення моделі високорівневою комунікації можлива.

Розширені уявлення. У разі мультиагентной системи агенти могут володіті уявленнямі НЕ только про Зовнішнє середовище и ее Реакції на їх Дії, а й про можливий реакцію зовнішнього середовища на Дії других агентів, а також про Можливі Дії других агентів.

Для моделювання уявлень агентів про можливі наслідки дій інших агентів відношення уявлень необхідно розширити, включивши в нього дії всіх агентів системи: $bel_{ag} \subseteq S \times ACS \times S$. Крім того, уявлення агента про можливі дії інших агентів, соціальні уявлення, описуються ставленням $sbel_{ag} \subseteq S \times ACS$, визначальним

можливі, з точки зору даного агента, дії системи для кожного з станів зовнішнього середовища. Безліч всіх можливих соціальних уявлень позначимо SBel(S,ACS) , $2^{S\times ACS}$.

Крім того, необхідно змінити сигнатуру функції поновлення уявлень агента таким чином, щоб вона враховувала не тільки дії, вчинені даними агентом, але і дії інших агентів системи, а також оновлювала соціальні уявлення агента:

br f_{ag} : $I_{ag,bel} \times Bel(S,ACS) \times SBel(S,AG) \times 2^{ACS} \times 2^{S} \rightarrow I_{ag,bel} \times Bel(S,ACS) \times SBel(S,AG)$. (1.2)

Таким чином, зміна уявлень агента визначається на підставі поточного стану уявлень $i_{ag,bel} \in I_{ag,bel}$, поточних уявлень $bel_{ag} \in Bel(S,ACS)$ і соціальних уявленнях $sbel_{ag} \in SBel(S,AG)$, а також доступною агенту інформації про вчинені дії $p_{ACS} \subseteq ACS$ і поточному стану зовнішнього середовища $pS \subseteq S$.

Розширення уявлень агента вимагає і розширення його бажань. У разі мультиагентной системи бажання агента представляються безліччю функцій-критеріїв виду $goal_{ag}$: $(2^{S} \times Bel(S,ACS) \times SBel(S,ACS))^* \rightarrow$ {completed, continue, failure}. Безліч всіх таких функцій позначимо як Goal(S,ACS). Функція поновлення бажань в цьому випадку набуде вигляду dr fag :

$$I_{ag,des} \times 2^{Goal(S,ACS)} \times 2^{S} \rightarrow I_{ag,des} \times 2^{Goal(S,ACS).}$$

Коаліції агентів. Для Досягнення загально и особістом цілей агенти об'єднуються в коаліції и діють спільно, об'єднуючи свои знання и возможности.

Ключовим поняттям при моделюванні поведінки мультиагентной системи є поняття коаліції. коаліція $C \subseteq AG$ є деяка підмножина агентів системи, що діють спільно для досягнення особистих і загальних цілей. Агенти, що входять в коаліцію C, діють спільно, довіряють один одному, обмінюються інформацією і координують свої дії. При цьому вони не довіряють агентам поза коаліцією C і не можуть ніяк вплинути на поведінку цих агентів. Коаліцію агентів, що не увійшли в C, назвемо коаліцією-антагоністом і позначимо C , AG \ C.

Коаліція агентів C = {ag1, . . . , agnC} _ AG формально може бути прийнята як єдиний інтелектуальний агент $ag^C = (S^C,A^C,env^C,I^C,refine^C,action^C)$, побудований наступим чином:

• S^C , S × (ACS ∪ {∅}) — безліч станів зовнішнього середовища для коаліції є декартовій твір безлічі станів зовнішнього середовища системи в цілому і безлічі дій системи або порожнім безліччю (для ідентифікації початкових станів).

• A^C , $A_{ag1} × ... × A_{agn}C$ — безліч дій коаліції є декартовій твір множин дій агентів коаліції.

• $env^C = \{(s^C,a^C,s'^C) ∈ S^C × A^C × S^C \mid ∃a^C ∈ A^C : s'^C|_{ACS} = a^C + a^C$ and $(s^C|_S,s'^C|_{ACS},s'^C|_S) ∈ env\}$ — зовнішнє середовище коаліції, в порівнянні з зовнішнім середовищем системи в цілому, має великий недетермінізмом за рахунок дій агентів, які не входять до коаліції. Крім того, частина стану зовнішнього середовища коаліції, відповідна діям системи, повинна однозначно визначатися діями.

• I^C , $I_{ag1} × ... × I_{agn}C$ — безліч внутрішніх станів коаліції є декартовій твір множин внутрішніх станів агентів коаліції.

• $refine^C(i^C,s^C)$, $(refine_{ag1}(i^C|I_{ag1},s^C|S,s^C|_{ACS}),...,refineagnC\ (i^C|I_{agnC}$, $s^C|_S,s^C|_{ACS}$ — функція оновлення стану коаліції є векторфункція, складена з функцій поновлення стану окремих агентів.

• $action^C(i^C)$, $(action_{ag1}(i^C|_{Iag1}),...,action_{agn}C\ (i^C|_{IagnC}))$ — функція прийняття рішення коаліції є вектор-функція, складена з функцій поновлення стану окремих агентів.

Крім того, для коаліції-агента можна визначити сприйняття зовнішнього середовища і ментальний стан, комбінуючи відповідним чином сприйняття і ментальні стану входять в коаліцію агентів.

1.2 Аналіз існуючих робіт з розробки мультиагентих систем

Історично першим методом реалізації мультіагентніх інтелектуальних систем є метод класної дошки, вперше застосований в системі HEARSAY [69],

призначеної для розпізнавання мови. Метод включає наступні основні архітектурні елементи:

• глобальну структуру даних, звану класною дошкою (blackboard);

• кілька паралельно працюючих носіїв знань, експертів (knowledgesource), які постійно бачать вміст класної дошки;

• центральний контролюючий механізм (control algorithm) або протокол, який визначає порядок доступу експертів до класної дошки.

В даному випадку експерти виступають як основні учасники процесу решення завдання. Вони не спілкуються між собою безпосередньо, а взаємодіють з контролюючим механізмом і поміщають інформацію на дошку. Експерти постійно сканують вміст дошки або отримують повідомлення про егоізмененіі. Спочатку на дошці формулюється завдання, яке необхідно вирішити. Якщо один з експертів знає спосіб розв'язання задачі або хоча б інформацію, яка, на його думку, може допомогти вирішенню завдання, він запитує контролюючий механізм отримання доступу на запис до класної дошки. Коли доступ буде отримано, експерт записує інформацію на дошку, і вона стає видимою всім іншим експертам.

Існує багато варіацій методу класної дошки. Зокрема, можна завести класну дошку для записів цілей і / або планів агентів, і в цьому випадку вона послужить зручним інструментом для вирішення проблеми узгодженості. Однак наявність централізованого контролюючого механізму і єдиної централізованої структури даних є одним з основних обмежень даного підходу і не дозволяє ефективно використовувати його в повноцінно розподілених інтелектуальних системах, таких як мультиагентні системи.

Інший метод був розроблений Дугласом Ленатом в його концепції істот.Ця концепція була спрямована на розробку методу, що дозволяє моделювати рішення проблеми спільнотою експертів, кожен з яких є фахівцем в певній предметній області. На відміну від методу класної дошки, Ленат запропонував метод, заснований на взаємодії експертів за допомогою питань і відповідей. При цьому і питання, і відповіді Ленат пропонував розсилати широкомовно. Істоти, в концепції

Лената, моделювали таких експертів. Для перевірки концепції істот Ленат розробив додаток PUP6 [4], де кожна істота моделювався на LISP у вигляді фрейму з двадцятьма сімома слотами (в термінології Лената частинами (parts)), які представляли питання, на яке істота могло відповісти. При приході повідомлення, істота порівнювало його зі своїми слотами, що могло призвести до широкомовної посилці нового повідомлення. Досвід, отриманий при розробці PUP6, Ленат використовував в своєму відомому проекті Artificial Mathematician (AM) [5].

Концепція істот краще підходить для моделювання агентів і мультиагентних систем, ніж класна дошка. Великим плюсом концепції є відсутність централізованого механізму управління і повноцінна распределенность. Однак істоти в концепції Лената не є самонавчального і не здатні до накопичення та аналізу досвіду. Набір їх знань фіксований, і вони не мають можливість їх розширювати.

Перші напрацювання концепції акторів [6, 7] з'явилися практично одночасно з роботами Лената по істотам. Концепція була заснована на об'єктно-орієнтованому підході до моделювання і могла розглядатися як метод проектування паралельних об'єктно-орієнтованих систем. На відміну від істот, концепція акторів донині активно розвивається як в теорії, так і на практиці в різних розподілених системах.

Система складається з набору акторів, які залишаються пасивними до тих пір, поки не отримають повідомлення. Коли повідомлення отримано, актор намагається вибрати відповідне повідомленням поведінку (behavior) (в ранніх роботах поведінку називалося сценарієм (script)). Реакція актора на повідомлення могла бути трьох видів:

- посилка нового повідомлення собі або іншим агентам;
- створення додаткових акторів;
- визначення заміщення (replacement) поведінки.

Повідомлення в системі надсилаються асинхронно, а кожен актор має унікальну адресу, таким чином, повідомлення розсилаються НЕ широкомовно, а конкретним одержувачам. Кожен актор має чергу повідомлень, в якій зберігаються

повідомлення, що прийшли, поки вони не буду оброблені. Визначення заміщення поведінки має на увазі створення нового актора, який успадкує адресу і черга повідомлень свого творця. Тобто в цьому випадку такі повідомлення в черзі, а так само ті повідомлення, що відправляться пізніше, будуть оброблені вже новим актором.

Великим плюсом концепції акторів є її високий паралелізм, обмежений виключно можливостями ЕОМ. Однак актора ще не можна вважати повноцінним агентом, так як він представляє досить просту обчислювальну одиницю, все ще не володіє можливостями до самонавчання.

Multi-Agent Computing Environment або MACE - повноцінна мультиагентна інтелектуальна система, описана Гессером [8]. Система складається з наступних п'яти компонент:

– набір прикладних агентів (application agents), які були основними обчислювальними одиницями системи;

– набір визначених системних агентів (system agents), які надавали сервіс користувачеві;

– набір засобів загального призначення, доступних все агентам (facilities);

– база даних з описами (description databases), яка містила описи агентів і могла призвести виконуваний код на основі опису;

– набір ядер (kernels), по одному на кожну фізичну машину, які забезпечували комунікацію і маршрутизацію повідомлень.

– Гессер виділив три аспекти діяльності агентів: вони містять знання, вони відчувають навколишнє середовище, вони виконують дії. Агенти MACE мали два типи знання: спеціалізовані (domain knowledge), орієнтовані на предметну область, і ознайомчі знання (acquaintance knowledge) - знання про інших агентів системи. Агент зберігав таку інформацію про інших агентів.

– Клас (class) - агенти організовувалися в групи, звані класами та ідентифікуються на ім'я класу.

– Ім'я (name) - кожному агенту присвоювалося унікальне в класі ім'я, таким чином, в рамках системи агента можна було ідентифікувати парою рядків (клас, ім'я).

– Роль (roles) - опис ролі, виконуваної агентом в своєму класі.

– Навички (skills) - безліч можливостей описуваного агента за відомостями описує агента.

– Мета (goals) - безліч цілей, яких описується агент хоче досягти за відомостями описує агента.

– Плани (plans) - уявлення описує агента про те, як описуваний агент буде досягати свої цілей.

Агенти відчували зовнішнє середовище за допомогою отримання повідомлень, а єдиний видимий результат діяльності агентів полягав в розсилці повідомлень, адресованих конкретному агенту, групі агентів або всім агентам.

Універсальність і гнучкість системи MACE дозволяла моделювати багато інших інтелектуальні системи. Зокрема, метод класної дошки моделювався за допомогою MACE досить легко.

Система MACE стала важливою віхою в розвитку розподілених інтелектуальних систем, і багато нововведень, в ній введені, активно застосовувалися в наступних розробках. Одним з найважливіших наслідків її розробки є початкова формалізація агентно-орієнтованої методології програмування.

На сьогоднішній день мультиагентні системи використовуються для розробки широкого спектра інформаційних систем, серед яких умовно можна виділити три основні класи:

Відкриті системи. Системи, структура яких може змінюватися в про-

процесі їх функціонування. Найбільший і відкрита на сьогоднішній день система - Інтернет. Соціальність і автономність агента дозволяють ефективно застосовувати його в якості елемента відкритої системи.

Складні розподілені системи. Найкращими сучасними методами боротьби зі складністю, є модульність і абстракція. Завдяки високому ступеню автономності, агент дозволяє зменшити кількість залежностей між частинами системи і, отже, спростити її проектування і реалізацію.

Інтерактивні системи. Більшість існуючих систем, незважаючи на графічний інтерфейс і потужну систему довідки, вимагають серйозних зусиль з боку потенційного користувача для їх освоєння. За допомогою агентів можна побудувати інтерактивну систему, яка буде не просто приймати і виконувати команди користувача, а активно і інтелектуально взаємодіяти з ним, прагнучи до досягнення загальних цілей.

У промисловості мультиагентні системи найбільш поширені в наступних областях:

Автоматизація управління складними системами. Область, в якій давно і ефективно застосовуються інтелектуальні агенти. Як приклади можна привести платформу ARCHON [9], систему управління виробництвом YAMS [10] і систему управління повітряним рухом OASIS [11].

Збір та обробка інформації. Агенти часто використовуються для реалізації систем, що збирають і обробляють інформацію з всесвітньої мережі Інтернет. Більшість сучасних пошукових машин реалізовано з використанням агентів.

Ігри. Сьогодні в комп'ютерних іграх противниками гравця людини часто стають гравці, реалізовані як інтелектуальні агенти.

Підходи до розробки мультиагентних систем можуть сильно відрізнятися в залежності від класу задач, для вирішення яких ця система призначена. Проте, існує кілька досить загальних методологій проектування МАС:

Gaia. Заснована на концепції ролі методологія запропонована Майклом Вулдріджом і Ніколасом Дженнінгсом [12]. Методологія пропонує розробку кількох типів моделей. На етапі аналізу системи розробляються модель ролей і модель взаємодій, які на етапі проектування уточнюються в моделях агентів, сервісів і обізнаності (acquaintance).

SODA. Методологія, орієнтована на розробку додатків для мережі Інтернет [13], ключовим поняттям якої є поняття завдання. Основна увага в даній методології приділяється соціальності агентів і пов'язаним з нею особливостям проектування.

Tropos. Методологія [14], основна увага якої приділяється організації мультиагентної системи. Основними поняттями методології є поняття актора (агента), цілі і залежності. фундаментом методології є система моделей і[15]

1.3 Формування мети задачі

Актуальність створення мультиагентної системи в корпоративній мережі підтверджується тим, що вартість і відповідальність прийняття рішень при роботі в корпоративній мережі весь час зростає, а час на прийняття цих рішень постійно зменшується. Тому особливо корисне застосування мультиагентих систем при керуванні небезпечними виробничими об'єктами з метою поліпшення і автоматизування процесів.

Про це свідчить масштаб і кількість техногенних аварій за минуле десятиліття, а також тяжкість наслідків після них. Крім цього, важливо враховувати інформацію про погіршення надійності та старіння обладнання, яке використовуються на виробництві. Так само, треба враховувати людський фактор при роботі зі складними системами, так як близько 70% аварій відбуваються з вини особи, яка приймає рішення.

В даний час розвиваються нові напрямки в інформаційних технологіях. Поява інтелектуальних систем, розподільних технологій і багато чого іншого змушують підприємства вдосконалюватися.

Можна стверджувати, що поліпшити ефективність мультиагентної системи при риботі в корпоративній мережі можна за допомогою переходу до систем накопичення знань і орієнтування на прогнозування.

Складність розробки мультиагентної системи для корпоративної мережі визначена тим, що з'являється необхідність створення складних баз знань і баз даних.

Виходячи з цього, виникає необхідність розробити таку мультиагенту систему, яка буде допомагати та полегшувати роботу особі, яка приймає рішення, а саме – системному адміністраторові, при роботі з великими об’ємами інформації у корпоративній мережі.

Мета роботи: Допомога та полегшення умов роботи для особи, яка приймає рішення при роботі в корпоративній мережі за допомогою розробки мультиагентної системи.

Завдання, які вирішуються в роботі:

Аналіз існуючих мультиагентних систем для полегшення роботи з великими об’ємами інформації в корпоративній мережі.

Розробка мультиагентної системи.

Аналіз мультиагентної системи.

Предложено разработать мультиагентую систему, взяв за основу и модернизировав под требования корпоративной сети многоагентную вычислительную среду (Multi-Agent Computing Environment или MACE) - полноценную мультиагентную интеллектуальную систему, описаную Гессером.

С целью показа основніх процессов, протекающих в корпоративной сети, разработана мультиагентая система поддержки принятия решений, позволяющая лицу, принимающему решение, допомагати та полегшувати роботу з великими об’ємами інформації.

1.4 Висновки

Проаналізовано поняття «агент», «інтелектуальний агент» та «мультиагента система». Визначено, що агентом є обчислювальна система, вміщена в зовнішнє

середовище, здатна взаємодіяти з нею, роблячи автономні раціональні дії для досягнення цілей. Визначено, що навідміну від звичайного агента інтелектуальний агент повинен мати наступні властивості:

- реактивність;
- проактивність;
- соціальність.

Введено визначення мультиагентної системи, яка представляється трьома критеріями:

- кінцевою множиною станів зовнішнього середовища;
- кінцевою множиною агентів, кожен з котрих представлений розширеною моделлю інтелектуального агента;
- функцією, що описує можливу реакцію зовнішнього середовища на дії всіх агентів системи.

Наведено та проаналізовано основні методи та різновиди мультиагентих систем:

- метод класної дошки і його варіації;
- метод концепції істот, разроблений Дугласом Ленатом;
- метод концепції акторів;
- повноцінна мультиагентна інтелектуальна система, опис Гессеріт.

Розглянуто, для яких основніх типів систем використовуються мультіагентіе системе:

- Відкриті системи;
- Складні розподілені системи;
- Інтерактивні системи.

Відзначено, що в промисловості мультиагентні системи найбільш поширені в наступних областях:

- Автоматизація управління складними системами;
- Збір та обробка інформації.
- Ігри.

Наведено підходи до розробки мультиагентних систем, що відрізняються в залежності від класу задач, для вирішення яких ця система призначена, а саме:

- Gaia;
- SODA;
- Tropos.

Поставлено мету розробити мультиагентую систему підтримки прийняття рішень для работи в корпоративній мережі, яка допоможе адміністратору приймати рішення при роботі з великими об’ємами інформації.

2 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ МУЛЬТИАГЕНТНОЇ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ В КОРПОРАТИВНІЙ МЕРЕЖІ.

Управління корпоративною розрахунковою мережею можна розглядати як безліч взаємозв'язаних завдань, вирішенням яких займаються спеціалісти різних областей знань інформаційних технологій, такі як: системні адміністратори, мережеві адміністратори, адміністратори баз даних, спеціалісти з інформаційної безпеки тощо.

І якщо для великих організацій існує можливість розділення цих обов'язків між кількома спеціалістами з профільною освітою, то для малих підприємств мережу обслуговує тільки системний адміністратор. У цьому випадку виникає проблема прийняття неоптимальних, суб'єктивних рішень, що може призвести до погіршення якості роботи корпоративної мережі.

Частково дану проблему вирішують такі класи інформаційних систем, як: системи управління мережами (англ. NMS - Network Management Systems), системи моніторингу, системи виявлення вторгнень (англ. IDS) і запобігання вторгненням (англ. IPS - Prevention of Intrusion). В той же час, деякі автори [16], [17] відзначають недостатню інтелектуальність існуючих в даний час систем.

Рішенням даної проблеми може стати застосування технологій штучного інтелекту до управління обчислювальними мережами.

Запропоновано розглянути використання агентного підходу для побудови системи підтримки прийняття рішень по управлінню корпоративною комп'ютерною мережею.

Важливою перевагою використання даного підходу є висока ступінь масштабування та адаптації кінцевої системи.

2.1 Функціональні вимоги до мультиагентої системи з управління корпоративною мережею

Міжнародний досвід у сфері управління мережами зв'язку узагальнено в рекомендаціях ITU - T X.700 і в близькому до них стандарті ISO 7498 - 4, які виділяють п'ять функціональних груп завдань управління (модель FCAPS):

- управління відмовами (англ. Fault Management) - це процес (або процеси) обробки відмов / аварійних сигналів в елементах мережі, збору і аналізу інформації про відмови;
- управління продуктивністю і надійністю (англ. Performance Management) - це процес (або процеси) збору та аналізу статистики мережевих елементів;
- управління конфігурацією і ім'ям (англ. Configuration Management) - це процес (або процеси) внесення змін в конфігурацію елемент мережі: додавання, усунення та модифікації;
- облік роботи мережі (англ. Account Management) - процес контролю за ступенем використання мережевих ресурсів і підтримування функції з нарахування оплати за це використання;
- управління безпекою (англ. Security Management) - комплекс заходів по здійсненню захисту мережі від несанкціонованого доступу.

В даному випадку під мережевим елементом розуміється об'єкт або обладнання, що використовується для надання послуг. До мережевих елементів відносяться мережеві пристрої: комутатори, маршрутизатори, робочі станції, сервера, а також працює на них програмне забезпечення. Модель FCAPS широко використовується при розробці систем управління мережами зв'язку та телекомунікаціями.

Провівши аналіз моделі FCAPS, можна зробити висновок про те, що лише частина описаних функцій може бути автоматизована. Наприклад, не можна повністю виключити людини при установці нового обладнання або заміни вийшов

з ладу. У той же час інтелектуальна обробка і аналіз даних про стан мережевих елементів і всієї мережі в цілому дозволили б істотно підвищити ефективність прийнятих рішень у всіх областях управління мережами.

Дані, необхідні для аналізу, можуть бути отримані як за допомогою стандартизованих протоколів управління, таких як SNMP, WMI, SSH,

Telnet, так і за допомогою спеціалізованих агентів моніторингу. Для аналізу даних застосовується метод нечіткого міркування на основі прецедентів

(Англ. Fuzzy CBR), який дозволяє узагальнювати досвід кількох експертів в процесі прийняття рішень. Також хотілося б відзначити важливість наступних функцій системи: моделювання і прогнозування стану КВС як результат прийняття рішень. Їх реалізація дозволяє значно зменшити кількість помилок в процесі прийняття рішень. Для вирішення задачі моделювання та прогнозування стану КВС необхідно використовувати мережевий симулятор ns3.

2.1.1 Агенти мультиагентої системи підтримки прийняття рішень з управління корпоративною мережею

До складу системи підтримки прийняття рішень з управління корпоративною мережею входять наступні основні типи агентів: інтерфейсний агент, агент зберігання, агент-складальник, агент- аналізатор. Також в системі існують два службових типи агентів: система управління агентами (англ. Agent Management System - AMS) і агент служби каталогів (англ. Directory Facilitator - DF). Останні два агента є частиною специфікації FIPA.

Інтерфейсний агент служить для організації взаємодії «користувач - система», в його завдання входить прийом і обробка команд користувача, а також уявлення відповіді системи. Система включає в себе три види інтерефейса:

- інтерфейс адміністратора;
- інтерфейс експерта;
- інтерфейс оператора.

Інтерфейс адміністратора призначений для управління самою системою і безпосереднього контролю над агентами. Інтерфейс експерта служить для перегляду і модифікації правил виведення системи підтримки прийняття рішень. Інтерфейс оператора призначений для безпосередньої роботи з СППР.

Агент зберігання даних покликаний забезпечувати інтерфейс доступу системи до зовнішніх джерел: файловим архівам, баз даних і т.д. основними з них, з якими працює система, є: СУБД MySQL, СУБД RRDTool, а також файлова система.

Агент-складальник отримує дані з мережевих пристроїв по протоколах SNMP, WMI, Telnet і SSH. Також цей агент здійснює пошук пристроїв в мережі.

Агент-аналізатор є ядром СППР, в його завдання входить інтелектуальна обробка зібраних даних з метою вироблення рекомендацій щодо поліпшення роботи обчислювальної мережі, моделювання результатів застосування рекомендацій, а також прогнозування роботи мережі. Всі основні агенти є багатокомпонентними системами. Так, агент зберігання даних об'єднує агентів, що працюють з конкретними джерелами даних. Такими є MYSQLDB, RRDDB, FILESYSTEM, що надають інтерфейси для доступу до інформації, що зберігається в СУБД MySQL, RRDTool і файлової системи відповідно. Набір даних агентів залежить від інформаційної структури підприємства. Інтерфейсний агент включає в себе агентів «Інтерфейс адміністратора», «Інтерфейс експерта» і «Інтерфейс оператора».

Агент-складальник складається з наступних компонентів: SNMPGetter, SSHGetter, WMIGetter, TelnetGetter і Scanner. Їх завдання: отримання інформації від мережевого обладнання, серверів і робочих станцій по протоколам SNMP, SSH, WMI, Telnet, а також пошук пристроїв в мережі.

До складу агента-аналізатора входять:

- агент пошуку рішень;
- агент моделювання;
- агент прогнозування;

- агент пошуку;
- агент усунення несправностей.

Агент пошуку несправностей перевіряє відповідність контрольованих параметрів нормі і в разі відхилення сигналізує про нього. Агент усунення несправностей виконує пошук комплексів організаційно-технічних заходів, повертає мережу в штатний стан.

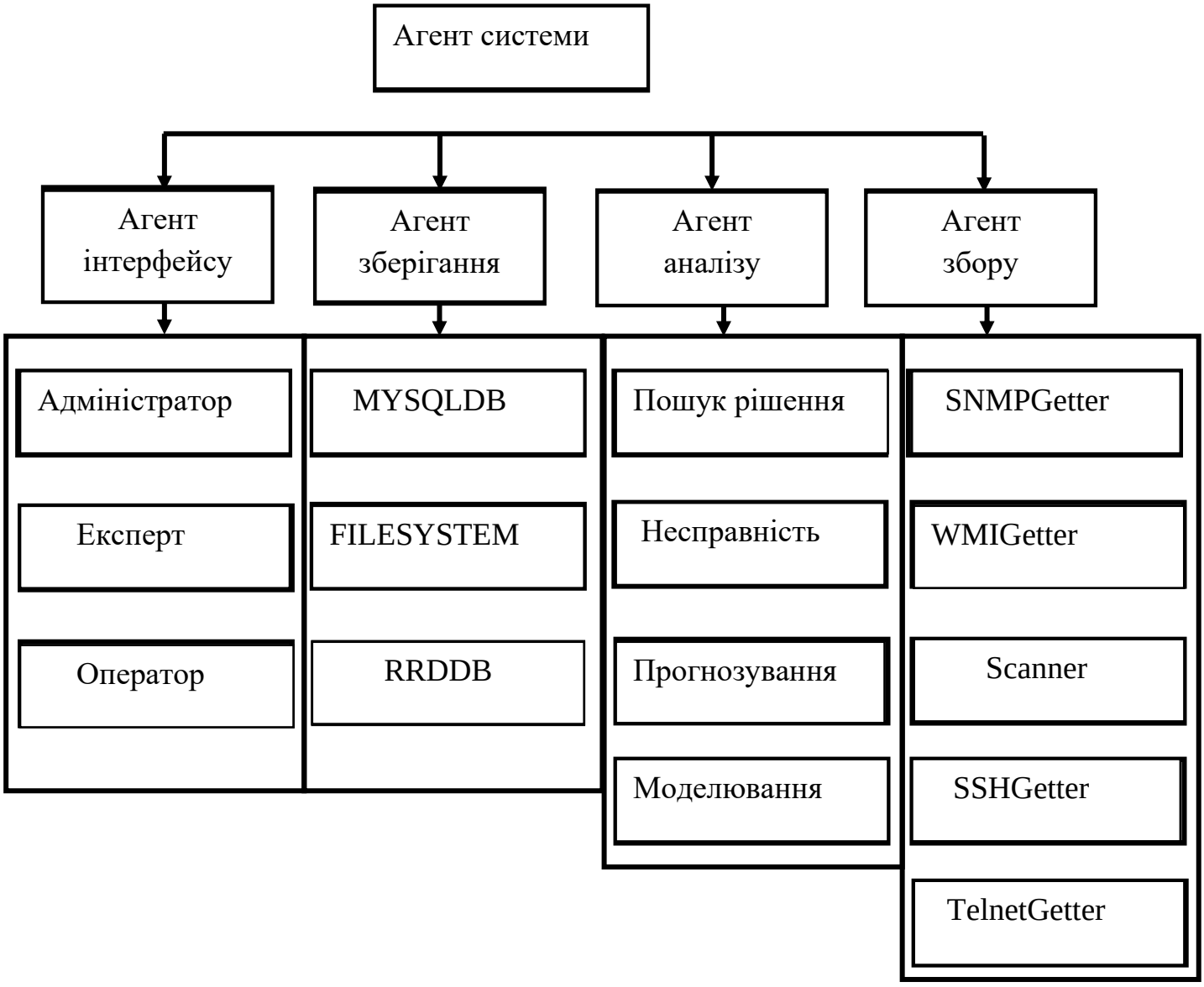

Рисунок 2.1 - Ієрархія основних агентів СППР з управління корпоративною мережею

Агенти моделювання і прогнозування покликані оцінити наслідки вживання заходів, запропоновані агентом усунення несправностей. Завданням агента

прийняття рішення є оцінка знайдених заходів щодо усунення несправностей і наслідків їх застосування.

Службові агенти AMS і DF необхідні для забезпечення функціонування платформи в цілому. Згідно зі специфікацією, FIPA AMS служить інтерфейсом для управління агентами і платформою в цілому, AMS зберігає актуальні списки агентів і контейнерів, які існують всередині системи. DF надає службу каталогів, використовувану агентами для пошуку один одного по шаблонах. DF зберігає відповідність між агентами і наданими ними службами.

Важливою перевагою використання агентного підходу є простота модифікації системи шляхом додавання нових агентів, які розширюють або уточнюючих функції вже існуючих. Склад агентів може бути легко змінений навіть після введення системи в експлуатацію. Наприклад, для того, щоб система мала можливість отримувати дані по протоколу sFlow, досить реалізувати відповідного агента.

2.1.2 Вибір архітектури та мови реалізації мультиагентої системи

В якості основи для побудови СППР з управління КВС послужила FIPA - сумісна Агентна платформа JADE [18] (Java Agent Development Framework). До складу даної платформи входять: бібліотека об'єктів для розробки многоагентних систем і самих агентів, Виконавча агентів, а також набір утиліт для управління платформою і налагодження. Важливими перевагами даної платформи є: відповідність стандартам FIPA, відкритий і вільно поширюваного вихідного тексту, доступність документації, реалізації платформи на мовах JAVA і C #, кроссплатформенность, активна спільнота розробників, велика кількість доповнень, які розширюють функціональність платформи.

Розробники JADE підтримують наступні редакції JAVA: J2SE, J2EE, PersonalJava і CLDS, що дозволяє використовувати дану платформу на різних

пристроях, починаючи від персональних комп'ютерів і закінчуючи мобільними телефонами з підтримкою MIDP. Основними сутностями, що складають основу JADE є наступні.

Агент - програмна сутність, здатна здійсняться в рамках платформи JADE.

Повідомлення - одиниця обміну інформацією всередині платформи. Взаємодії агентів всередині платформи будується виключно на основі обміну повідомленнями. Сервіс передачі повідомлень використовує технологію JAVA RMI для передачі інформації між двома вузлами системи або виклик методів всередині одного вузла.

Контейнер - Виконавча агентів. На кожному вузлі системи можуть бути запущені кілька контейнерів. У кожному контейнері може одночасно існувати кілька агентів.

Платформа - логічне об'єднання декількох контейнерів. При старті платформи автоматично створюється головний контейнер (main container), в якому запускаються агенти AMS і DF.

Агентна платформа JADE відповідно до стандартів FIPA має гібридну P2P архітектуру, тобто в MAC існують два типи вузлів: загального призначення і супервузли. Особливість супервузлів полягає в тому, що вони дозволяють управляти системою, до них відносяться агенти AMS і DF. Вузли загального призначення реалізують функції MAC.

2.1.3 Агентна платформа JADE для розробки мультиагентої системи

JADE повністю написана на мові програмування Java з використанням таких просунутих можливостей як Java RMI, Java CORBA IDL, Java Serialization і Java Reflection API.

Вона спрощує розробку мультиагентних систем завдяки використанню FIPA-специфікацій і за допомогою ряду інструментів (tools), які підтримують фази

виправлення помилок (debugging) і розгортання (deployment) системи. Агентна платформа може поширюватися серед комп'ютерів з різними операційними системами, і її можна конфігурувати через віддалений GUI-інтерфейс. Процес конфігурації цієї платформи досить гнучкий: її можна змінити навіть під час виконання програм, для цього необхідно просто перемістити агентів з однієї машини на іншу. Єдиною вимогою цієї системи є установка на машині Java Run Time 1.2. комунікаційна архітектура пропонує гнучкий і ефективний процес обміну повідомлення, де JADE створює чергу і управляє потоком ACL-повідомлень, які є приватними для кожного агента. Агенти здатні звертатися до черги за допомогою комбінації декількох режимів своєї роботи: блокування, голосування, перерва в роботі і зіставлення з еталоном (що стосується методів пошуку). На даний момент в системі використовується Java RMI, event-notification, і IIOP, але легко можна додати і інші протоколи. Також передбачена можливість інтеграції SMTP, HTTP і WAP. Більшість комунікаційних протоколів, які вже визначені міжнародною спільнотою розробників агентних середовищ, доступні і можуть ілюструватися на конкретних прикладах після визначення поведінки системи і її основних станів. SL і онтологія управління агентами також імплементовані разом з підтримкою певних користувачем тематичних мов, а також онтології, які можуть бути імплементовані і зареєстровані агентами і використані системою. З метою істотного розширення працездатності JADE, передбачена можливість інтеграції з JESS і Java-оболонкою CLIPS.

JADE використовується низкою компаній і академічних груп. Серед них можна виділити такі відомі: BT, CNET, NHK, Imperial College, IRST, KPN, University of Helsinky, INRIA, ATOS та багато інших.

Згідно зі специфікацією FIPA повідомлення містить кілька полів, такі як: тип повідомлення, відправник, отримувач, відповідь, зміст, мову, кодування, онтологія, протокол і т.д. У JADE в якості змісту повідомлення може виступати рядок, об'єкт JAVA або спеціальна структура, сформована на основі онтології. Причому останній варіант є кращим, з точки зору теорії багатоагентних систем. Онтологія

визначається як сукупність понять предметної області і відносин між ними. Агентна платформа JADE дозволяє створювати окремі онтології для організації взаємодії між агентами і, крім того, містить вбудовані онтології для управління агентами і платформою в цілому. На рис. 2 показаний фрагмент онтології, розробленої для СППР з управління КВС, яка описує відносини між елементами мережі. Основними відносин, використовуваними в даному фрагменті, є:

- відношення "is - a" - це відношення «узагальнення - деталізація», ставлення більшою чи меншою абстракції;
- відношення "has - one" і "has - many" - це відношення «ціле - частина», відношення приналежності.

При створенні мультиагентних систем на основі платформи JADE необхідно враховувати, що вона заснована на архітектурі Hybrid - P2P, тобто агенти всередині платформи можуть вільно взаємодіяти один з одним безпосередньо, але при цьому існує координаційний центр, керівник цієї платформою. Таким центром є головний контейнер з агентами AMS і DF. З точки зору відмовостійкості, ця особливість є важливою, так як головний контейнер являє собою єдину точку відмови всієї платформи. Для вирішення цієї проблеми розробники JADE пропонують такі засоби:

- реплікація основного контейнера;
- зберігання каталогу DF в реляційної базі даних.

Можливість реплікації основного контейнера передбачає створення кількох його копій разом з агентом AMS. Створені репліки синхронізують дані між собою. Таким чином, при виході з ладу головного контейнера його функції візьме на себе одна з реплік.

За замовчуванням агент DF зберігає дані про сервіси в оперативній пам'яті. Збереження даних каталогу DF в реляційної базі даних дозволяє при виході з ладу основного контейнера з агентом DF автоматично запустити нового агента DF в новому основному контейнері. При цьому новий агент DF восстановит каталог з бази даних.

Використання програмних комплексів управління обчислювальними мережами дозволяє підвищити ефективність обслуговування корпоративних обчислювальних мереж малих підприємств. Застосування агентноорієнтированной парадигми до розробки подібних систем є ефективним інструментом підвищення їх інтелектуальності. Агентна платформа JADE дозволяє розробникам зосередитися на логіці взаємодії агентів, не відволікаючись на підтримку середовища їх існування. Онтологію системи наведено на рис. 2.2.

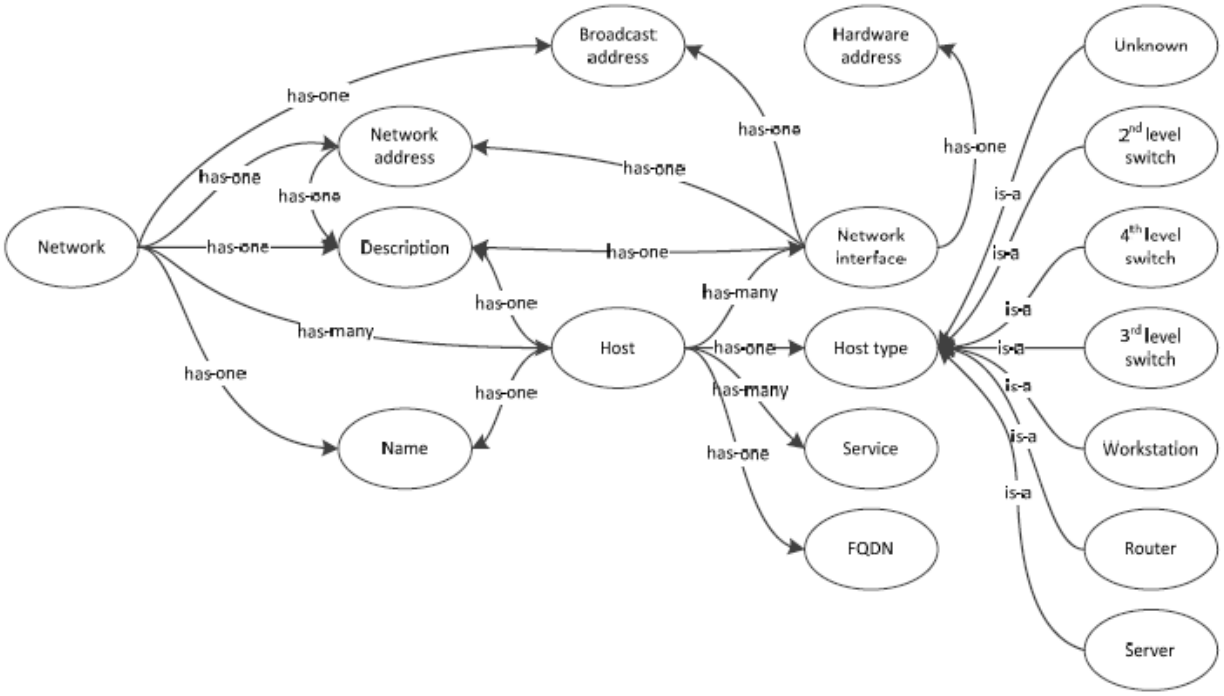

Рисунок 2.2 - Фрагмент онтології системи

2.2 Розробка мультиагентної системи підтримки дистанційного навчання в мережі

В якості експерименту розглянуто мультиагентну систему підтримки дистанційного навчання - одну з найпростіших моделей застосування агентів в системі підтримки дистанційного навчання (СПДН).

Нехай, є три агента: "слухач" який взаємодіє з "викладачем" використовуючи можливості "адміністратора". При цьому функціональність кожного агента така ж як і в реальному процесі навчання. Наприклад, одним із сценаріїв такої взаємодії

може бути: "слухач" здає тест перевірки знань і відправляє "викладачеві" для перевірки. "викладач" перевіряє тест і передає результати тестування "адміністратору".

Поняття автономності є природним і дуже важливим для систем дистанційного навчання. Зрозуміло, що слухач буде віддавати перевагу індивідуального стилю навчання. Всі дії як слухача так і вчителі, адміністратора наперед передбачити неможливо. Тому персональний помічник однієї з груп повинен мати можливість самостійної оцінки ситуації, бажань, уподобань і т.п. і на їх основі: породжувати нові сценарії спілкування; робити перехоплення ініціативи діалогу; мати здатність самонавчання з допомогою використання зворотного зв'язку між учасниками спілкування; кооперувати дії кількох представників груп агентів на рішення якоїсь спільної задачі.

Використання агентів дає можливість спростити перехід від реального світу, реального навчального процесу, до його відображення у віртуальному світі. Зберігаючи ту ж саму модель навчання, що і в реальному університеті, ми робимо перший крок для збереження якості дистанційного навчання. Міграція агентів може підтримуватися не тільки між постійн приєднаними до мережі вузлами, а й між мобільними платформами, як підключаються до постійної мережі на деякі проміжки часу і можливо по низькошвидкісних каналах. Клієнт приєднується до постійної мережі на короткий проміжок часу з мобільної платформи, відправляє агента для виконання завдання і від'єднується; потім клієнт приєднується до іншого вузла мережі і забирає результати роботи агента. Інший варіант - сервер, на який повинен перемістится агент, приєднується до мережі, а потім від'єднується. В цьому випадку агент повинен вміти перемістится на такий сервер, який тимчасово приєднується, і повернутися в постійну мережу.

Отже, перспективними галузями використання мобільних агентів в КСПДО бачаться: навігація і перегляд, отримання інформації зі сховищ, сортування і класифікація, фільтрація; нагадування, програмування, диспетчеризація (scheduling), підтримка радами; тренінги, орієнтація в предметі, надання допомоги,

пошук нової інформації; противник в іграх, партнер в іграх. Природним є тут і використання базової метафори: інтуїція агента.

Ідея агентного інтерфейсу - спроба створити середовище, в якому були б використані навички, які відносяться до досвіду спілкування - спілкування з іншими людьми. Звідси і антропоморфізм або персоналізація, яка часто називається серед бажаних особливостей агента [Городецький В.І., 1996]. Однак, в чисто текстово-діалогової середовищі "партнер" по діалогу швидше уявний, ніж достатньо відчутний. І можливість представити його вимагає як деяких зусиль, так і досить складно організованого і підготовленого свідомості користувача - того інтерфейсу, який "в голові". А якщо ми реалізуємо Агентно метафору в сенсуального (зокрема, візуальної) середовищі, ми можемо зробити партнера-агента як завгодно детально візуалізувати, анімованим, озвученим - навіть відчутним (при наявності відповідної ВР-периферії) і інтуїтивно зрозумілим нам, якщо в його реакціях ми будемо згадувати архетипи вираження відносин між людьми та іншими оточуючими нас живими істотами.

Коли мова йде про комунікативні середовища, концепція агента доповнюється тим, що, делегуючи йому завдання, ми доручаємо йому діяти від нашого імені, приймаючи тим самим, прямо або побічно, в повному або обмеженому обсязі, відповідальність за його дії. Агент - це завжди чийсь агент.
Згідно загального визнання, базовим стандартом до розробки КСПДО останнім часом стала система IDEAL (The intelligent distributed environment for active learning)

2.2.1 Система IDEAL

Система підтримує персоніфіковану взаємодію користувачів з навчальною системою, допомагає в автоматичній оцінці результатів, і забезпечує прості у використанні засоби розробки інтерактивних мультимедійних додатків. Система

також пропонує новий підхід до організації змісту курсу, який заснований на чітких компонентах інструкції, названих lecturelets. Вони створені для спеціальної, орієнтованої на замовника інтерактивної презентації предметів вивчення. Вони замкнуті, автономні і можуть з легкістю бути пристосованими, інтегрованими в велику кількість навчальних систем.

IDEAL орієнтована на активне використання сучасних засобів Інтернет, Веб, цифрових бібліотек, і мульти-агентних технологій. IDEAL підтримує відкриту архітектуру з відкритими стандартами інформаційних технологій і може забезпечувати велику кількість спеціальних дій. IDEAL складається з декількох спеціалізованих агентів. В системі кожен студент користується персональним унікальним агентом, який надає йому персональний профіль, який включає загальні поняття, вивчення стилів, розрахований на його інтереси, і ін.

Персональний агент спілкується з іншими агентами системи за допомогою різноманітних каналів зв'язку. онлайн курс супроводжується набором агентів та агентів курсів і навчальних блоків.

Агенти курсів надають матеріали курсів і спеціальні навчальні технології курсу.

Навчальний агент взаємодіє зі студентом, виконуючи функції інтелектуального викладача курсу. Кожен навчальний агент отримує матеріали курсу і специфічні технології курсу у агента курсу, а потім намагається навчити студента даного матеріалу в найбільш відповідною для нього формі, базуючись на знаннях і інтересах студента. Навчальні агенти володіють і пропонують такі когнітивні знання і навички, як розуміння мови, спілкування, генерування природної мови, навчання, соціальні аспекти. Ці вміння роблять взаємодія студента з системою простим і природним, з використанням звичайних форм спілкування. Мультимедійні презентації, такі як графіка і анімація, допомагають краще зрозуміти складні моменти навчання.

У груповому навчанні, інструктори та студенти систематично працюють разом, виконуючи спільні завдання. Існують три основні аспекти в онлайновому

навчанні в групі: в залежності від встановлених комунікаційних каналів (Один-одному, один-багатьом, і багато до багатьох); від того, як знайти інших людей, які мають схожі інтереси; як зробити наочним і зрозумілим контекст.

Навчання в групі, запропоноване системою IDEAL складається з набору персональних частин і агентів курсу. Персональна частина складається з користувача і його власних агентів. Кожен з персональних агентів користувача може отримати профіль користувача і допомогти йому в зборі, обміні і перегляді інформації. Агент курсу надає інформацію, знання або контекст у взаємодії групи і діє як посередник між студентами. Вони можуть збирати профілі користувачів і створювати, продукувати інформацію про навчальну групу.

Моделювання студента є одним з найбільш важливих місць в КСПДО.

Існує велика кількість підходів до створення, генерування моделей студента. Більшість з них дуже складні і вимагають значних витрат (мережі Байеса, теорія наочності Демпстера-Шафера). В системі IDEAL, моделювання студента виводиться з показників продуктивності, використовуючи мережу Байєса. Міра того, наскільки добре вивчені певні навички, представлена як ймовірність розподілу за рівнями, таким як початківець (novice і beginning), середній (intermediate), високий (advanced) і експерт. Грунтуючись на цій моделі, імовірнісний розподіл рівнів знань, заданих показників продуктивності може бути задано лінійної тимчасової оцінкою. Спираючись на теорію Байеса і на допущення, що показники продуктивності незалежні, виведемо такі умовні ймовірності рівнів знань.

Планування розкладу - це одна з ключових компонент в інтелектуальній системі домашнього навчання. планування розкладу можливо уявити у вигляді подвійного процесу: знаходження важливої теми і її вибір. Студент готовий вчити тему лише якщо він достатньо підготовлений. Якість засвоєння теми обумовлюється можливостями студента в доступі до шаблонів по матеріалу курсу. В системі IDEAL студент має можливість дозволити навчального агенту вибрати наступну тему або зробити це самостійно. В обох випадках студент повинен

досягти досить високої оцінки готовності за тему. Якщо наступна тема вибирається навчальним агентом, він вибере ту, у якій найбільша оцінка готовності. Якщо студент вирішить вибрати наступну тему, йому надається діаграма залежності тим з пропозиціями, яку тему повторити і / або які нові теми вивчати.

В системі IDEAL використовується ефективний електронний засіб для представлення, обробки і відображення навчальних матеріалів. В його основі лежить використання активних документів XML (eXtensible Markup Language) для організації та передачі матеріалів курсу. Матеріали курсу розбиваються на маленькі компоненти, які відповідають lecturelet щодо предмета вивчення. lecturelet є "розумними" документами XML, а саме документами XML, які включають не тільки склад, а також код Java. Lecturelet можуть динамічно збиратися, щоб відповідати темам курсу в залежності від індивідуального прогресу студента. Цей підхід стає можливим завдяки правильному комбінуванню використання чотирьох технологій: XML, фрейми, агенти і сховище даних.

2.2.2 Інструментарій розробки мультиагентої системи підтримки дистанційного навчання

Розробники IDEAL експериментували з кількома розподіленими об'єктно-орієнтованими середовищами для створення мультиагентних систем, включаючи такі відомі, як Java Remote Method Invocation (RMI), JATLite, JavaSpace. Вони підкреслюють, що Java technologies, Java RMI пропонує проміжний мережевий рівень, який дозволяє JAVA об'єктів, які розподілені на розподілених сайтах ефективно спілкуватися, використовуючи звичайні методи запитів. Це надійно, має високу продуктивність і працює на багатьох обчислювальних платформах. JATLite дозволяє користувачам швидко створювати програмних агентів, які ефективно спілкуються через Internet і підтримують мобільних агентів. Він надає різноманітні функціональні можливості агентам, включаючи реєстрацію на Agent Message

Router (AMR), використовуючи ім'я і пароль, підключення / відключення до / від Internet, асинхронну посилку / отримання повідомлень і передачу файлів по FTP. Проблема з поточною реалізацією така, що програмне забезпечення не дуже стабільне і іноді повністю зависає. JavaSpace підтримує ефективний розподіл спілкування і обмін даними і надає простий, виразний і потужний інструмент, який полегшує традиційний тягар побудови розподіленого програмного забезпечення. JavaSpace -дуже нова, повільна система, яка ще не до кінця розроблена.

2.2.3 Прототип реалізацій мультиагентної системи підтримки дистанційного навчання

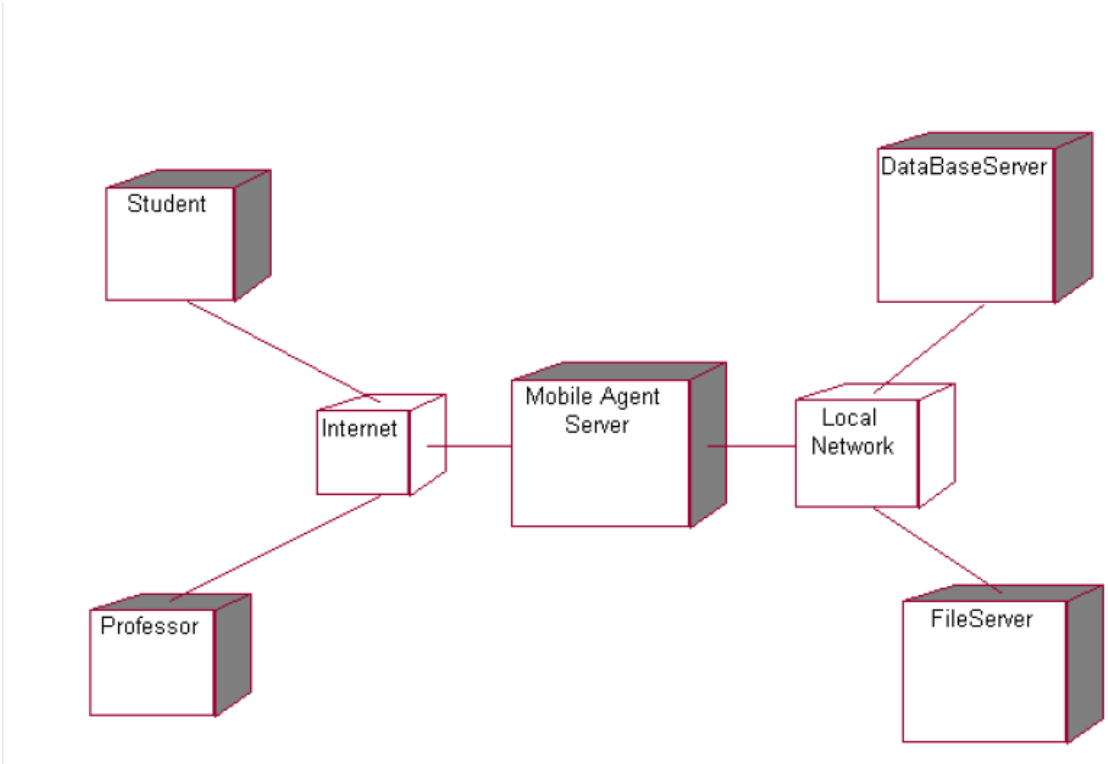

Рисунок 2.3 – Модель агентної системи підтримки дистанційного навчання співробітників

Нехай за допомогою UML була розроблена модель КСПДО агентного типу, діаграма архітектури якої показана на рисунку 2.3, де:

- Student - комп'ютер студента, який використовує мобільний агент для навчання;

- Professor - комп'ютер професора, який використовує сервер дистанційного навчання (мобільного агента) для перевірки і надання нових завдань;
- Internet - група пристроїв, що надають доступ до глобальної мережі Інтернет;
- Mobile Agent Server - сервер мобільних агентів;
- Lockal Network - пристрої локальної мережі, що забезпечують зв'язок серверів;
- DataBase Server - сервер БД (наприклад, MS SQL Server 2000, Oracle9i, DB / 2 Sybase), який містить інформацію про предмети, студентах і т.п .;
- File Server - файловий сервер, зберігає всі матеріали і контрольні роботи.

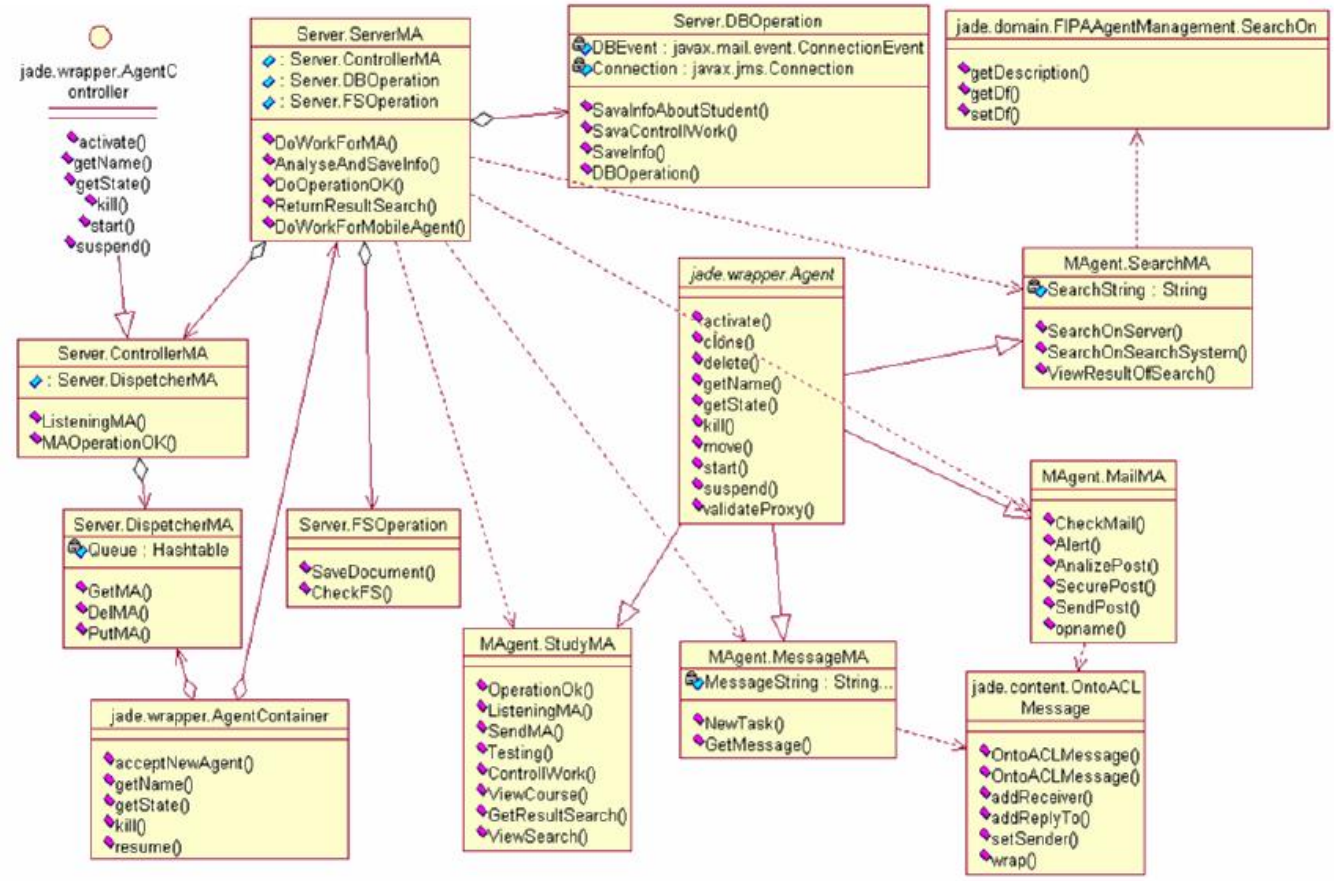

Рисунок 2.4 - Діаграма фізичної структури системи

- Server.ServerMA - клас, який використовує атрибути класів Server.DBOperation, Server.FSOperation, Server.ControllerMA і використовує їх для своєї роботи;
- Server.DBOperation- клас, який реалізує операції роботи з базою даних і містить такі методи як SaveInfoAboutStudent (), SaveControlWork (), SaveInfo () і об'єднується з операцією агрегації з класом Server.ServerMA. Клас містить атрибути, що визначають статус з'єднання агентів з сервером мобільних агентів;
- Server.ControllerMA - клас, який реалізує управління чергою, яку обслуговує Server.DispetcherMA;
- Server.DispetcherMA - клас, який планує пріоритет в черзі агентів MAgent.StudyMA, MAgent.MessageMA, MAgent.MailMA, MAgent.SearchMA. Відзначимо, що найвищий пріоритет має агент, який передає повідомлення MAgent.MessageMA; потім йде MAgent.StudyMA, який реалізує навчання студента, його тестування і спілкується як зі студентом, так і з викладачем; наступним по пріоритету йде MAgent.SearchMA, що виконує пошук матеріалів на сервері дистанційного навчання або в Інтернет;
- jade.wrapper.Agent - абстрактний клас, пов'язаний операцією успадкування з класами мобільних агентів. Він відноситься до стандартної бібліотеці класів JADE. Містить безліч методів, відповідальних за операції з агентами;
- jade.wrapper.AgentController - інтерфейс, що визначає ті методи, які дозволено контролювати JADE-агенту. Він містить в собі такі методи: activate () - активує агента, який з якихось причин припинив свою роботу; getName () - отримує ім'я платформи агента; getState () - отримує стан агента; kill () - знищує агента; start () - запускає агента; suspend () - перериває роботу агента на деякий час;
- jade.wrapper.AgentContainer - цей клас є proxy класом, який дозволяє звертатися до контейнера JADE-агента. acceptNewAgent () - метод, що дозволяє додавати агента в контейнер; getState () - повертає екземпляр стану платформи; resume () - активує Агентно платформу.

- jade.content.OntoACLMessage - забезпечує роботу з ACL- повідомленнями, які проходять згідно специфікації FIPA 2000 "FIPA ACL Message Structure Specification".

- jade.domain.FIPAAgentManagement.SearchOn - цей клас забезпечує операцію пошуку. Він дозволяє також DF-агенту (фасилітатору) робити запит іншому агентові-фасилітатору на пошук інформації.

Діаграма взаємодії класів агента навчання (Sequence Diagram) при пошуку даних на сервері або в глобальній мережі Інтернет показана на рисунку 2.5.

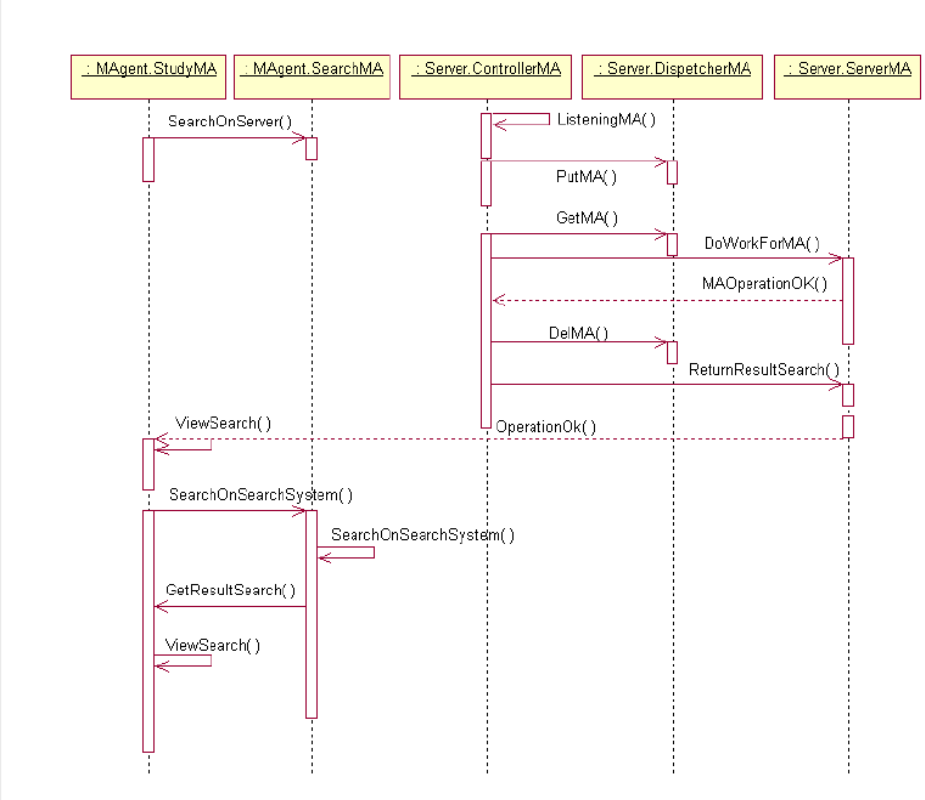

Рисунок 2.5 - Діаграма взаємодії класів агента навчання

Цю діаграму можна розбити на дві, але через те, що в ній беруть участь майже однакові класи, було б зручно об'єднати їх в одну з змісту, оскільки описується процес пошуку даних. Агенту навчання надійшло завдання знайти інформацію на

сервері. Використовує метод класу агента пошуку MAgent.SearchMA - SearchOnServer. Контролер агентів викликає метод диспетчера PutMA (), який визначає пріоритет і ставить агента в чергу. Server.ControllerMA викликає метод сервера DoWorkForMA (), виповнюється пошук і надсилається повідомлення, якщо операція завершилася успішно - MAOperationOK (). Диспетчер видаляє агента з черги. Контролер викликає метод сервера - повернути результати пошуку ReturnResultSearch (), які передаються агенту навчання разом з повідомленням про успішне завершення операції. Процес пошуку в мережі Інтернет аналогічний, з тією різницею, що для нього не використовуються дані з сервера мобільних агентів. Разглянуто рисунок 2.6 опису Sequence Diagram.

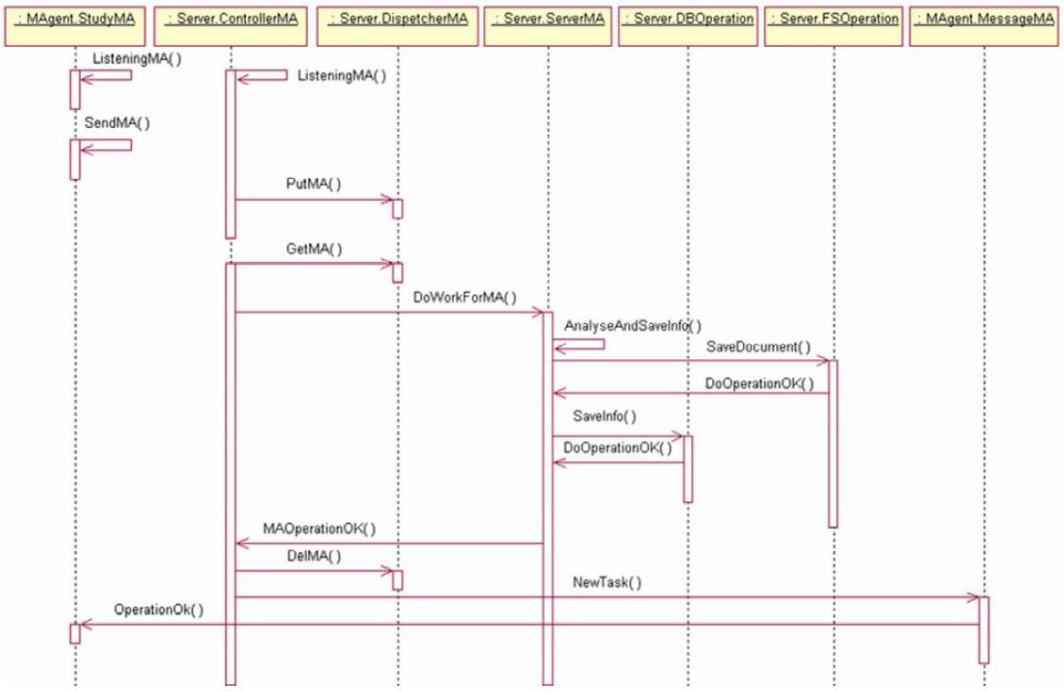

Рисунок 2.6 - Діаграма взаємодії класів агента навчання з контролером, диспетчером і сервером агентів

Клас StudyMA викликає функцію ListeningMA () і посилає агента.

Клас ControllerMA викликає функцію ListeningMA (), отримує агента і викликає метод класу диспетчера агентів DispetcherMA, який ставить пріоритет даного агента в черзі.

Контролер аналізує запит агента і викликає метод класу ServerMA - DoWorkForMA (). ServerMA аналізує і зберігає інформацію, отриману від агента, і викликає метод класу FSOperation - збереження документа, отриманого від агента SaveDocument ().

Клас FSOperation посилає серверу повідомлення, що операція пройшла успішно. Відповідна інформація зберігається в базі даних і клас DBOperation передає повідомлення серверу DoOperationOK (). Контролер отримує повідомлення від сервера агентів про успішну операцію і викликає метод класу диспетчера для видалення агента з черги DelMA (). ControllerMA викликає функцію класу MessageMA, для отримання нового завдання. MessageMA посилає повідомлення агенту, що операція успішна і можна робити інше завдання, яке надійде.

Діаграма взаємодії класів Агента повідомлень, Агента навчання, Mail-агента і Агента пошуку показана на рисунку 2.7

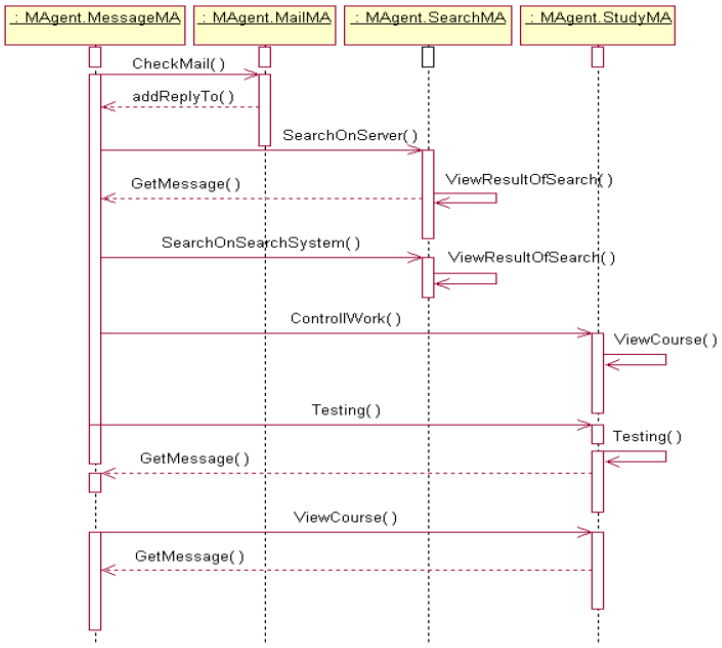

Рисунок 2.7 - Діаграма взаємодії класів агента повідомлень, агента навчання, mail-агента і агента пошуку

Від Агента повідомлень MAgent.MessageMA надходить запит на перевірку

електронної взаємодії. Клас MAgent.MailMA реалізує метод CheckMail () і посилає повідомлення про результати його роботи addReplyTo () агенту повідомлень. Клас MAgent.MessageMA викликає метод сервера SearchOnServer (). Агент пошуку MAgent.SearchMA переглядає результати пошуку ViewResultOfSearch (), і робить аналіз інформації, що надійшла. Агент повідомлень отримує повідомлення про результати пошуку GetMessage (). Клас MAgent.MessageMA викликає метод агента навчання MAgent.MessageMA, який здійснює, наприклад, перевірку контрольної роботи ControllWork (). Агент навчання здійснює перевірку і дає рекомендацію переглянути курси, в вправах до яких були допущені помилки ViewCourse (). Так само здійснюється процес тестування. Також можна просто послати повідомлення із запитом переглянути будь-який курс, який вас цікавить. В процесі взаємодії ці агенти використовують також стандартні методи агентной архітектури JADE, а саме, методи класу jade.content.OntoACLMessage.

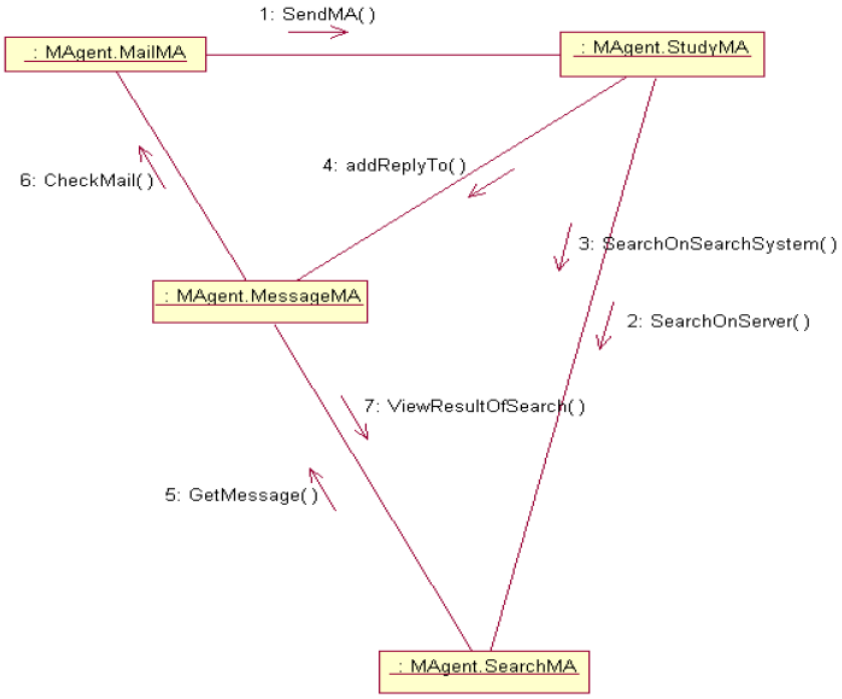

Рисунок 2.8 - Діаграма обміну повідомленнями між агентами пошуку, навчання, повідомлень та mail

Діаграма обміну повідомленнями між агентами пошуку, навчання, повідомлень, mail показана на рисунку 2.8.

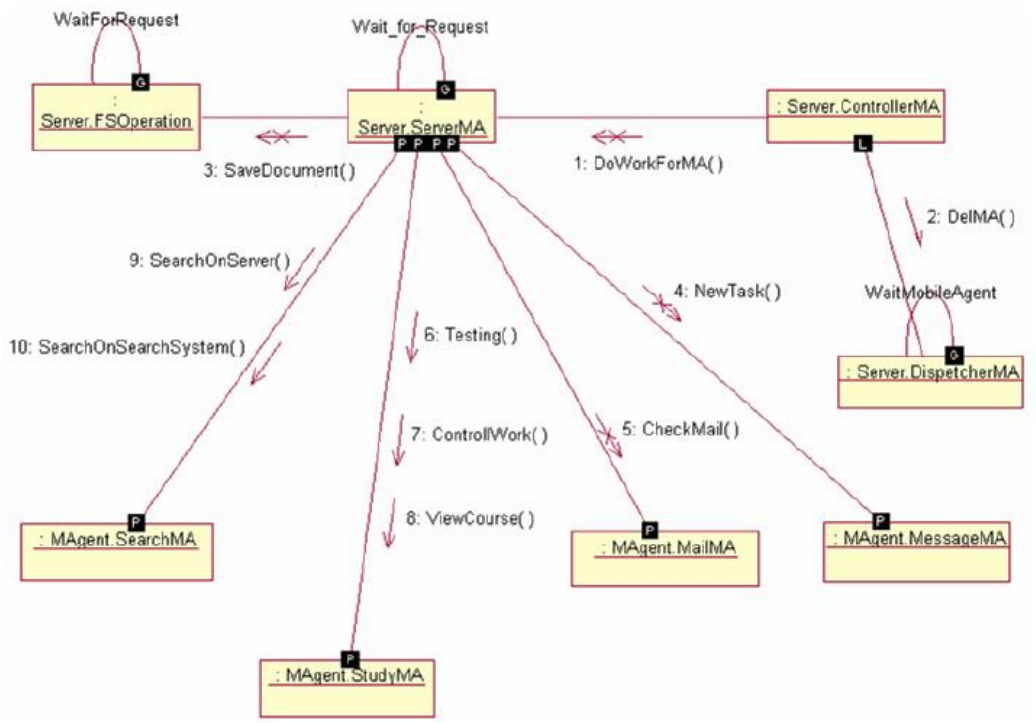

Рисунок 2.9 - Діаграма обміну повідомленнями між сервером мобільних агентів і агентами, а також контролером, диспетчером і файл-сервером

2.2.4 Аналіз мультиагентної системи підтримки дистанційного навчання

Проаналізовані як теоретичні, так і практичні особливості використання агентних технологій в дистанційному навчанні. Більшість систем дистанційного навчання, існуючих сьогодні в різних університетах України, майже не розглядаються комплексно як складні програмні системи. Тому цілком природним є виникнення потреби створення саме такої моделі, яка давала б цілісний погляд на процес побудови ефективних систем дистанційного навчання.

Ця спроба реалізована за допомогою агентної архітектури, яка є новітньою, дуже перспективною технологією завдяки своїм інтелектуальним властивостям, можливістю оперування знаннями. Для процесу дистанційного навчання важливим

є той факт, що агент здатний виконувати роль експерта, а також навчатися. Модель побудована з урахуванням основних характерних рис дистанційного навчання - гнучкості, модульності, паралельності, охвату, технологічності. Отримано такі нові наукові результати: досліджено систему дистанційного навчання як мультиагентну систему; розроблена модель дистанційного навчання на базі агентної архітектури JADE (Java Agent Development Environment); побудовано модель дистанційного навчання в середовищі проектування Rational Rose Professional Edition на мові UML (Unified Modeling Language); згенеровано каркас програмного коду системи мовою програмування Java.

Дана система відображає основні процеси, що трапляються при роботі з корпоративною мережею, тому є якісним фундаментом для використання здобвутих навичок при розробці мультиагентої системи в корпроативній мережі.

2.2.5 Висновки

В якості експерименту проведено дослідження та розробка мультиагентної системи підтримки дистанційного навчання. Спочатку виведено основні функціональні вимоги до мультиагентої системи з управління корпоративною мережею взагалі. Виділено основні агенти мультиагентої системи підтримки прийняття рішень з управління корпоративною мережею.

Оглянуто архітектуру та мову реализації мультиагентої системи з управління корпоративною мережею. Зважено особливості використання агентної платформи JADE при розробці мультиагентої системи з управління корпоративною мережею. Проведено дослідження мультиагентної системи підтримки дистанційного навчання в мережі з використанням системи IDEAL.Надано та обрано інструментарій розробки мультиагентої системи підтримки дистанційного навчання в мережі. Наведено прототип реалізацій мультиагентної системи підтримки дистанційного навчання в мережі. Проведено аналіз мультиагентної системи підтримки дистанційного навчання в мережі.

3 РОЗРОБКА МУЛЬТИАГЕНТНОЇ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ В КОРПОРАТИВНІЙ МЕРЕЖІ

3.1 Архітектура моделі мультиагентної системи підтримки прийняття рішень в корпоративній мережі

Так як в корпоративній мережі існує велика кількість процесів (виявлення та контроль відмов, сканування мережі, перевірки TCP, перевірки HTTP тощо), що працюють одночасно, а звідси виходить, що і об’єми інформації теж необхідно обробляти одночасно, бо вони можуть залежати один від одного, прийнято рішення розробляти модель мультиагентної системи підтримки прийняття рішень(СППР).

У даній моделі мультиагентної системи завдання розділяється між агентами рівномірно, і кожен агент виконує певну частину обов'язків. У такій моделі кожен агент виконує певну роль, складність вирішення задачі для агента визначають його можливості.

Кожен з агентів в такій моделі має кілька характеристик:

- Всі агенти незалежні, хоча б частково.
- У жодного агента немає уявлення про всю систему.
- Жоден агент не керує всією системою

Зазвичай СППР працює через адміністратора, роль якого виконує особа, яка приймає рішення. В мультиагентній СППР застосовуються програмні та інтелектуальні агенти, тому в рамках такої системи можливо створити взаємодію між людьми і роботами.

На підставі аналізу виділено наступні функції системи.

- Виявлення пристроїв.
- Моніторинг мережі.
- Зберігання даних моніторингу.
- Відображення даних моніторингу.

• Управління відмовами.

• Повідомлення про події.

Для вирішення перерахованих завдань виділено кілька типів агентів:

А1 - агент інтерфейсу.

А2 - агент адміністрування (управління).

А3 - агент формування звітів.

А4 - агент зберігання даних.

А5 - агент моніторингу.

Агенти взаємодіють один з одним тільки за допомогою передачі повідомлення, яке, в свою чергу, є найменшою одиницею передачі даних.

Базова архітектура моделі мультиагентонї СППР представлена на Рисунку 1.

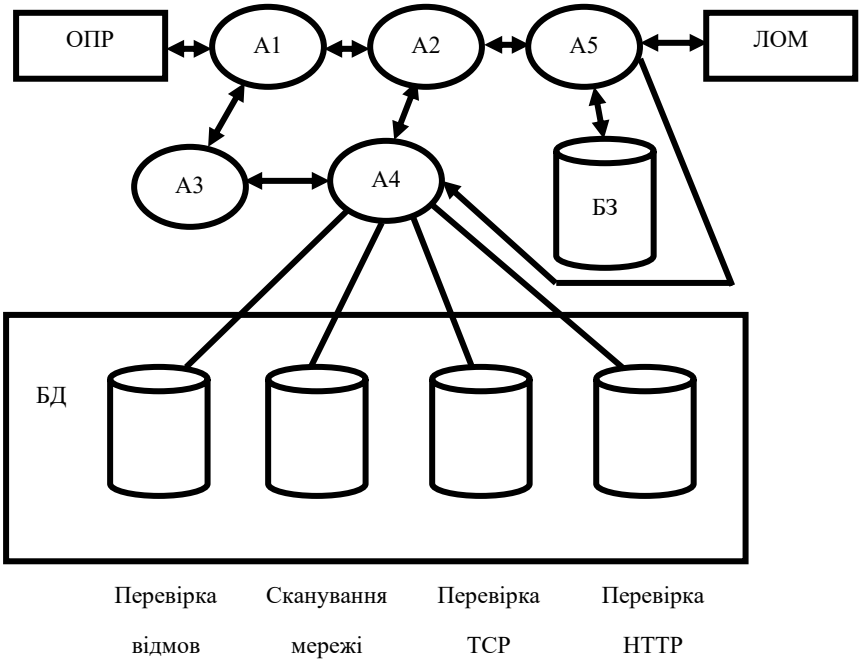

Рисунок 3.1 - Базова архітектура моделі мультиагентної СППР.

3.2 Інструмент для розробки мультиагентної системи підтримки прийняття рішень в корпоративній мережі

Основним засобом розробки обрано мережі Петрі.

Мережі Петрі - це математичний апарат, який допомагає моделювати динамічні дискретні системи.

Мережа Петрі - це двочастковий орієнтований граф, який складається з переходів і позицій. В такому графі вершини одного типу не можуть бути з'єднані, а в позиціях розташовуються мітки, які здатні рухатися по мережі.

Подія - це спрацьовування переходу, коли мітки з виходу переходять на вхід. Деякі події можуть відбутися миттєво, а деякі різночасно - це залежить від умов системи. Ці мережі створено для роботи з моделюванням систем з паралельно взаємодіючими компонентами.

Мережі Петрі мають особливі властивості:

- Число міток в позиції не може бути більше ніж певне число;
- Число міток, які можуть бути в позиції, не менш одного;
- Можливість переходу мережі з одного стану в інший;
- Перехід може спрацьовувати, якщо є можливість.

3.3 Концептуальна модель мультиагентної системи підтримки прийняття рішень в корпоративній мережі

При створенні СППР в основі лежать підходи, такі як: прийняття рішень, вилучення даних, відображення даних, побудова систем. Дані підходи засновані на зв'язку між людиною і машиною. Цьому відповідає концептуальна модель на рис. 3.2.

Блоки аналізу проблем і прийняття рішень включають в себе процедури і методи, що дозволяють сформулювати поставлену проблему за допомогою баз даних (БД), моделей (БМ) і знань (БЗ), проаналізувати можливості її рішення і

отримати результат. У СППР включаються також кошти для отримання даних і знань, побудови моделей і маніпулювання ними.

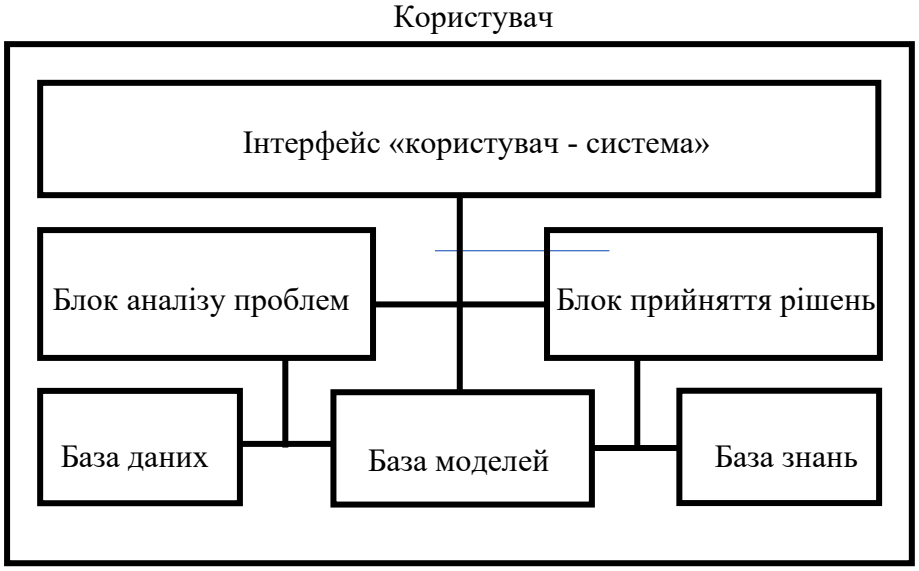

Рисунок 3.2 - Концептуальна модель СППР

При проектуванні використано мультиагентну СППР на основі нечіткого виведення.

Нечітке виведення займає центральне місце в нечіткій логіці і системах нечіткого управління. Процес нечіткого виведення являє собою деяку процедуру або алгоритм отримання нечітких виведень на основі нечітких умов або передумов з використанням понять нечіткої логіки. Цей процес поєднує в собі всі основні концепції теорії нечітких множин: функції приналежності, лінгвістичні змінні, нечіткі логічні операції, методи нечіткої імплікації і нечіткі композиції.

Системи нечіткого виведення призначені для перетворення значень вхідних змінних процесу управління у вихідні змінні на основі використання нечітких правил продукцій. Для цього системи нечіткого виведення повинні містити базу правил нечітких продукцій і реалізовувати нечіткий вивід висновків на основі посилок або умов, представлених у формі нечітких лінгвістичних висловлювань.

Нечітким лінгвістичним висловлюванням вважаються висловлювання наступних видів:

Вислів "β є α", де (β - найменування лінгвістичної змінної, α - її значення, якому відповідає окремий лінгвістичний терм з базового терм-множини Т лінгвістичної змінної β.)

Висловлення " β є ∇α", где ∇ - модифікатор, відповідний таким словами, як: "ДУЖЕ", "БІЛЬШ АБО МЕНШ", "БАГАТО БІЛЬШЕ" і іншим, які можуть бути отримані з використанням процедур даної лінгвістичної змінної.

Складові висловлювання, освічені з висловлювань видів 1 і 2 і нечітких логічних операцій в формі зв'язок: "І", "АБО", "ЯКЩО-ТО", "НЕ".

Розроблено наступну базу правил для реалізації мультиагентної СППР:

- Якщо при скануванні мережі виявлені відмова порту, то перезапустити службу, відповідальну за цей порт;
- Якщо при передачі файлів виявлений відмова порту, то перезапустити службу, відповідальну за цей порт;
- Якщо при передачі файлу відсоток помилки більше 5%, то знизити навантаження;
- Якщо при скануванні порту TCP результат відкритий або з'єднання прийнято, то можна надсилати та отримувати файл;
- Якщо при скануванні порту TCP результат закрито, заборонено або не слухає, то передавати або приймати файли заборонено;
- Якщо при скануванні порту TCP результат заблокований, відфільтрований, то від хоста не надходило відповіді, скористайтеся іншим;
- Якщо прийшов запит на протокол HTTP то, послати відповідь;
- Якщо при відправці запиту не прийшов результат, то послати повторно.

3.4 Структурна модель мультиагентої системи підтримки прийняття рішень в корпоративній мережі

А1-агент інтерфейсу (програмний). Агент контролю використовується для організації взаємодії користувача і системи. До функцій, що реалізовуються цим агентом, відносяться:

- управління об'єктами системи;
- відображення даних моніторингу;
- уявлення користувачеві звітів з аналізу даних моделювання, вироблених рекомендаціях.

В рамках взаємодії з користувачем агент адміністрування формує завдання і передає їх на виконання іншим агентам.

А2 - агент адміністрування (інтелектуальний). Його функцією є вироблення комплексу організаційно-технічних заходів, спрямованих на усунення відмов в мережі або оптимізацію її роботи на підставі даних моніторингу, моделювання, а також знань, закладених в системі експертами (рис.3.3).

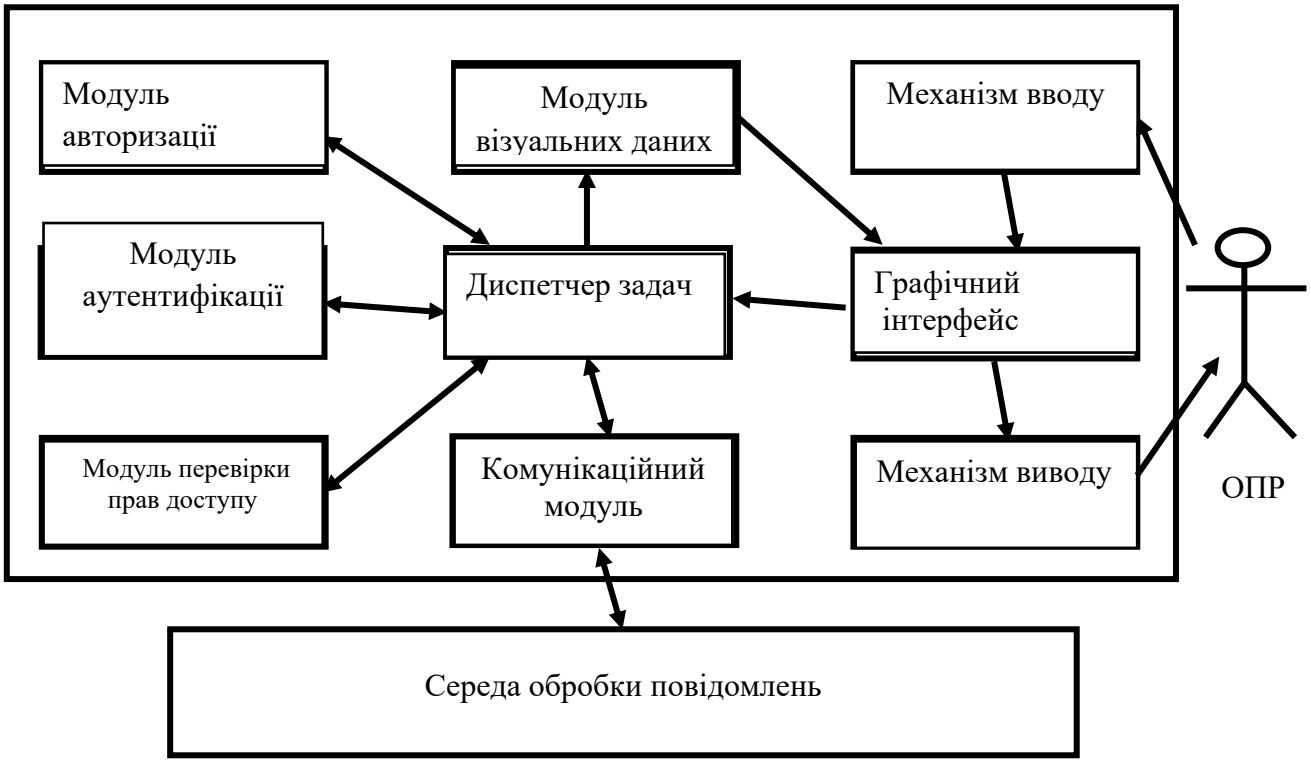

Рисунок 3.3 - Архітектура агента адміністрування системи.

А3 - агент звіту (програмний). До завдань агента формування звітів входить підготовка звітів на підставі даних, отриманих від агентів зберігання даних і агента пошуку рішення(рис 3.4).

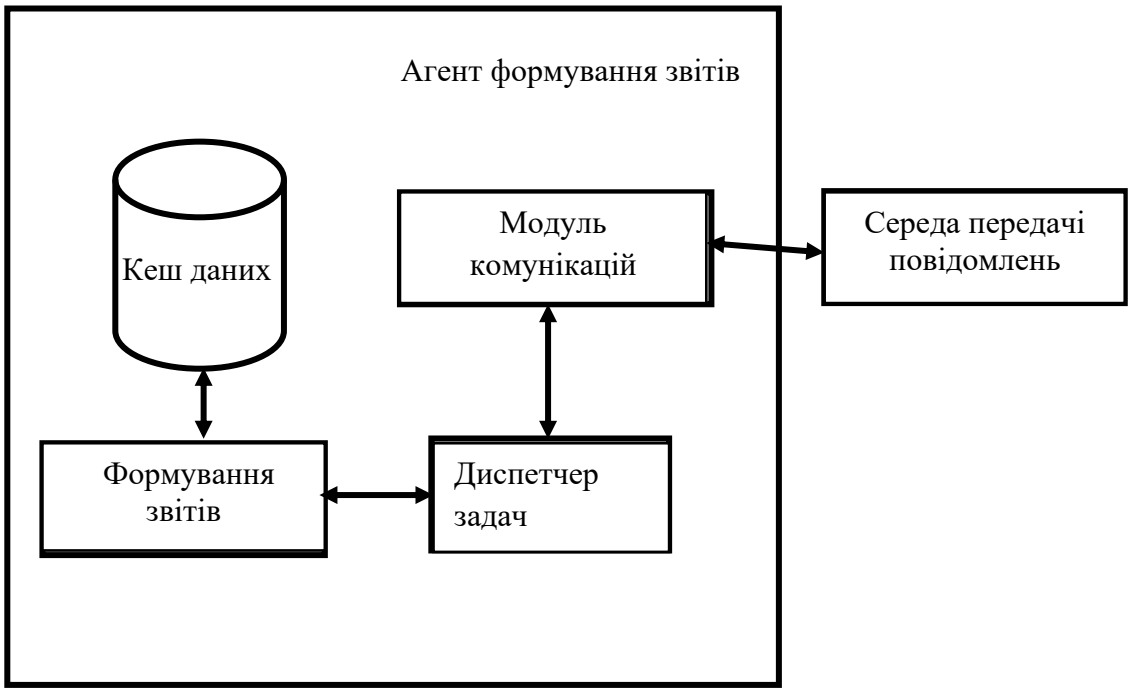

Рисунок 3.4 - Архітектура агента формування звітів

А4-агент зберігання даних (програмний). Агент зберігання даних використовується для зберігання і надання за запитом результатів моніторингу контрольованих параметрів елементів мережі, облікових записів користувачів, списку об'єктів управління. Для підвищення надійності і гнучкості даний агент може реалізовувати доступ до кількох баз даних(рис.3.5).

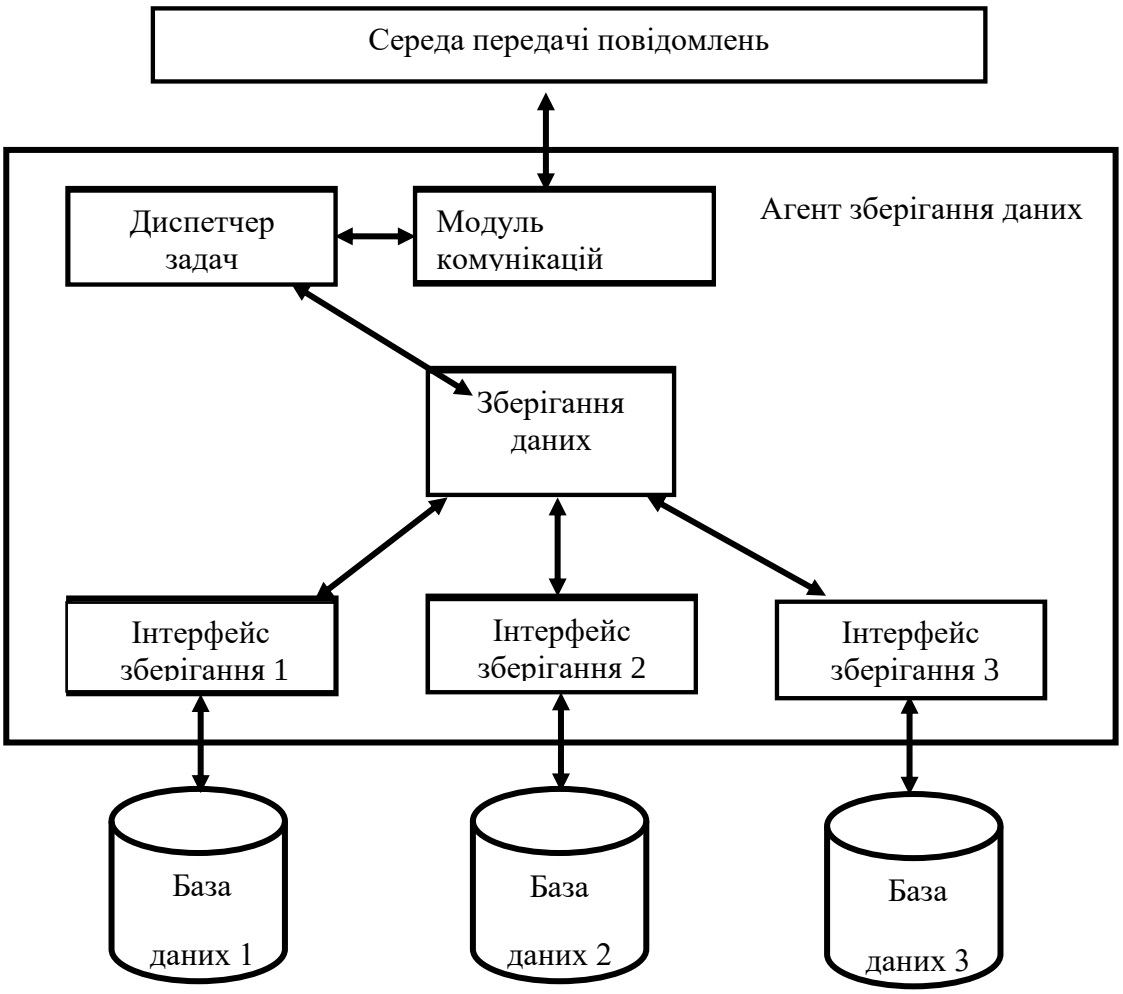

Рисунок 3.5 - Архітектура агента зберігання даних

А5-агент моніторингу (програмний). До завдань агента виявлення відносяться: збір і контроль значень спостережуваних параметрів елементів мережі; оповіщення про відхилення спостережуваних параметрів; передача управляючих впливів на елемент мережі. Зібрані дані передаються агентам зберігання даних(рис.3.6).

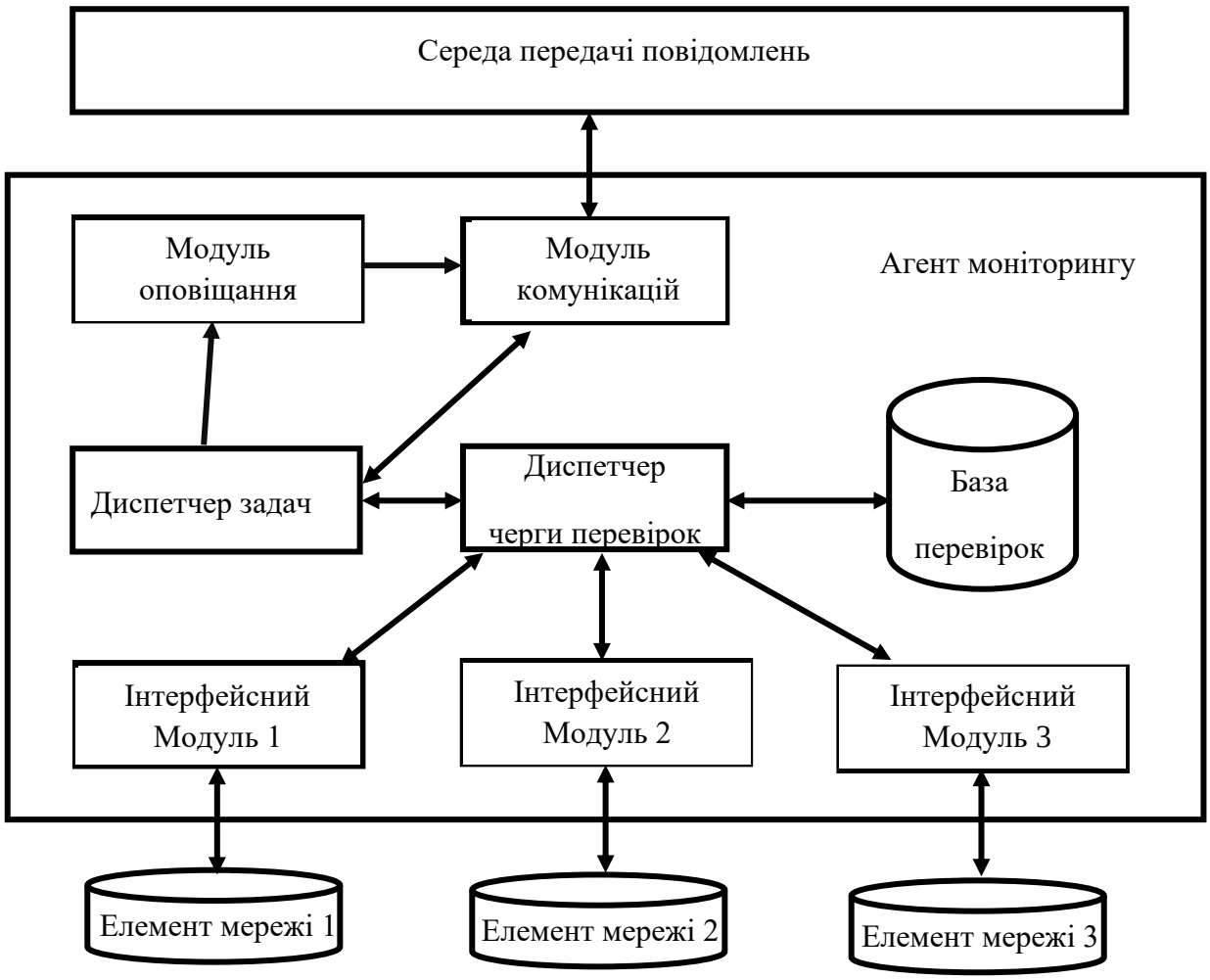

Рисунок 3.6 - Архітектура агента моніторингу

3.5 Модель мультиагентної системи підтримки прийняття рішень в корпоративній мережі

Розроблено модель мультиагентної СППР на базі мереж Петрі.

Схема мультиагентної системи на базі мереж Петрі представлена на рисунку 3.7. Варто відзначити, що при створенні системи, агенти і пов'язані з ними підсистеми представлені однією позицією.

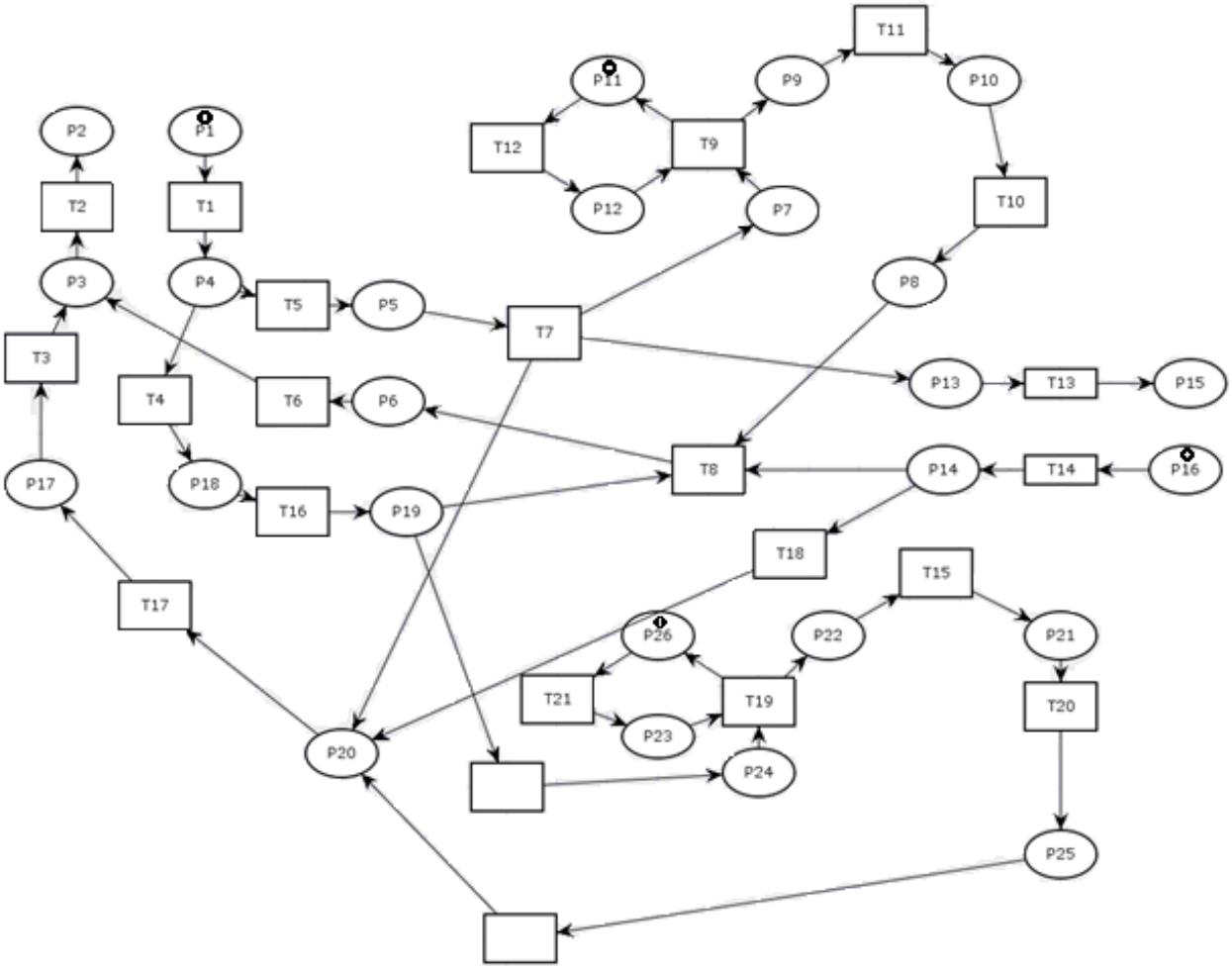

Рисунок 3.7 - Модель мультиагентної СППР

Введено наступні позначення:

- P1, P2 - особа, яка приймає рішення;
- P3, P4 - агент інтерфейсу;
- P17, P18 - агент формування звітів;
- P19, P20 - агент зберігання даних;
- P13, P14 - агент моніторингу;
- P15, P16 - локально обчислювальна мережа;
- P7 - P12 - база правил;
- P5, P6 - агент адміністрування;
- P21-P26-база даних;
- T1 - ОПР передає відповідь ЛОМ або запит на звіт;
- T2 - ОПР приймає сигнал від ЛОМ або звіт;

- T3, T17 - дані про звіт;
- T4, T16 - запит на звіт;
- T5, T13 - відповідь на запит від ЛОМ;
- T6, T14 - запит від ЛОМ;
- T7 - передача даних;
- T8 - запит на дані;
- T9, T10, T11, T12 - база правил;
- T15, T19-T21 - база знань;
- T18 - дані з ЛОМ.

В ході роботи було прийнято рішення поліпшити схему шляхом прискорення передачі сигналу від ОПР до агенту моніторингу(рис.3.8).

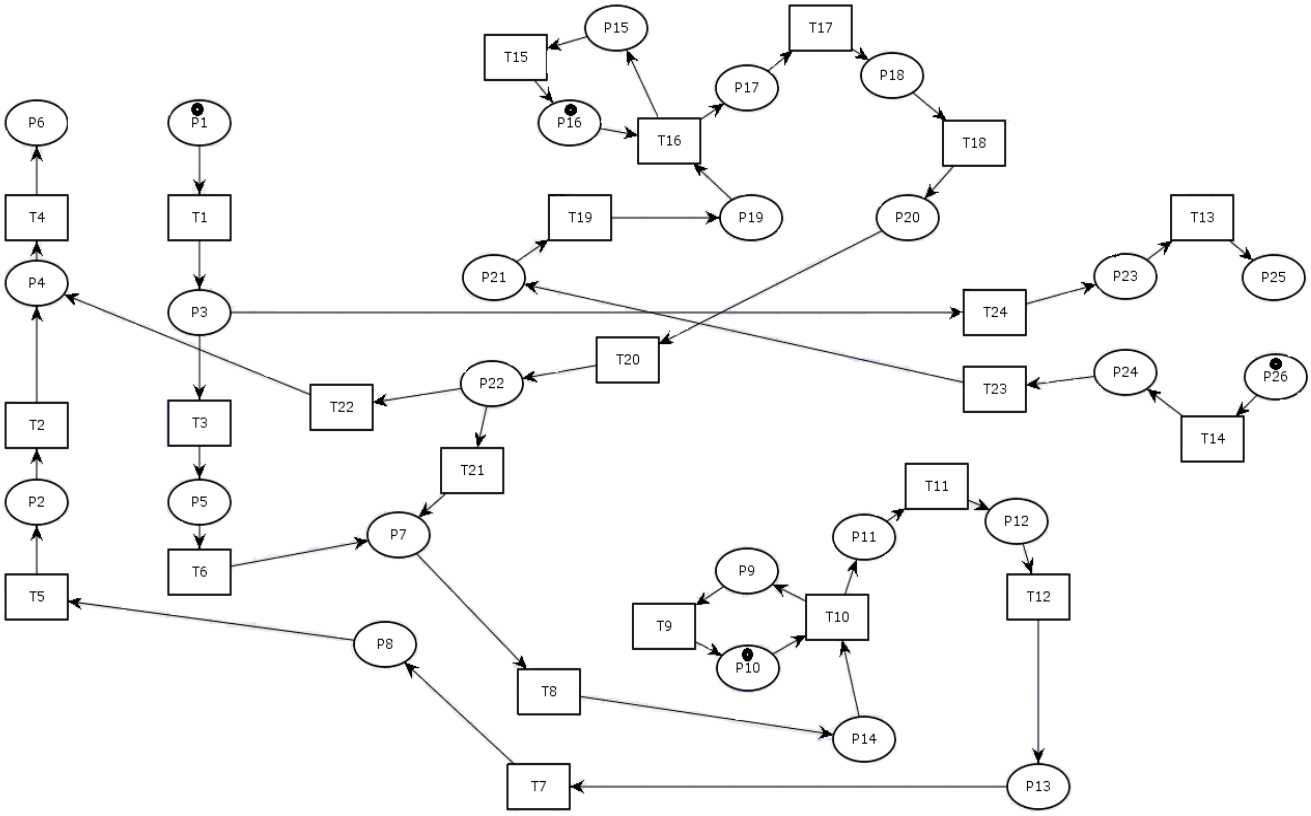

Рисунок 3.8 - Поліпшення схеми шляхом прискорення передачі сигналу від ОПР до агенту моніторингу

- P1, P6 - особа, яка приймає рішення;
- P3, P4 - агент інтерфейсу;
- P2, P5 - агент формування звітів;
- P7, P8 - агент зберігання даних;
- P23, P24 - агент моніторингу;
- P25, P26 - локально обчислювальна мережа;
- P15 - P20 - база правил;
- P21, P22 - агент адміністрування;
- P9-P14-база даних;
- T1 - ОПР передає відповідь ЛОМ або запит на звіт;
- T4 - ОПР приймає сигнал від ЛОМ або звіт;
- T2, T5 - дані про звіт;
- T3, T6 - запит на звіт;
- T7 - передача даних з бази правил;
- T8 - передача даних в базу правил;
- T13 - відповідь на запит від ЛОМ;
- T14 - запит від ЛОМ;
- T15-T18- база правил;
- T9-T12 - база знань;
- T18 - дані з ЛОМ;
- T19, T23 - перевірка даних через базу правил;
- T20 - дані перевірені через базу правил;
- T21 - відправка даних на зберігання в базу знань;
- T22 - сигнал ОПР про порушення бази правил;
- T24 - подача сигналу безпосередньо від ОПР до ЛОМ.

Таблиця 3.1- Позиції

Позиція	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15
Всього відпрацьовано	10	10	10	11	5	10	13	11	13	14	13	13	12	13	10

Таблиця 3.2- Переходи

Перехід	T15	T16	T17	T18	T19	T20	T21	T22	T23	T24
Всього .	10	10	10	10	10	10	8	2	10	5

Дана зміна дозволяє точно знати, скільки відповідей і запитів на звіт послала ОПР. Для моделювання вхідні позиції збільшили до 10 міток, а у вихідні помістили 10 міток. Система відпрацювала 30 кроків. Згідно з результатами моделювань надіслано 5 відповідей на запит (перехід T24) і 5 запитів на звіт (позиція P5). Через базу правил пройшли всі 10 міток.

Для покращення роботи бази правил, її вдосконалили до такого вигляду(рис.3.9):

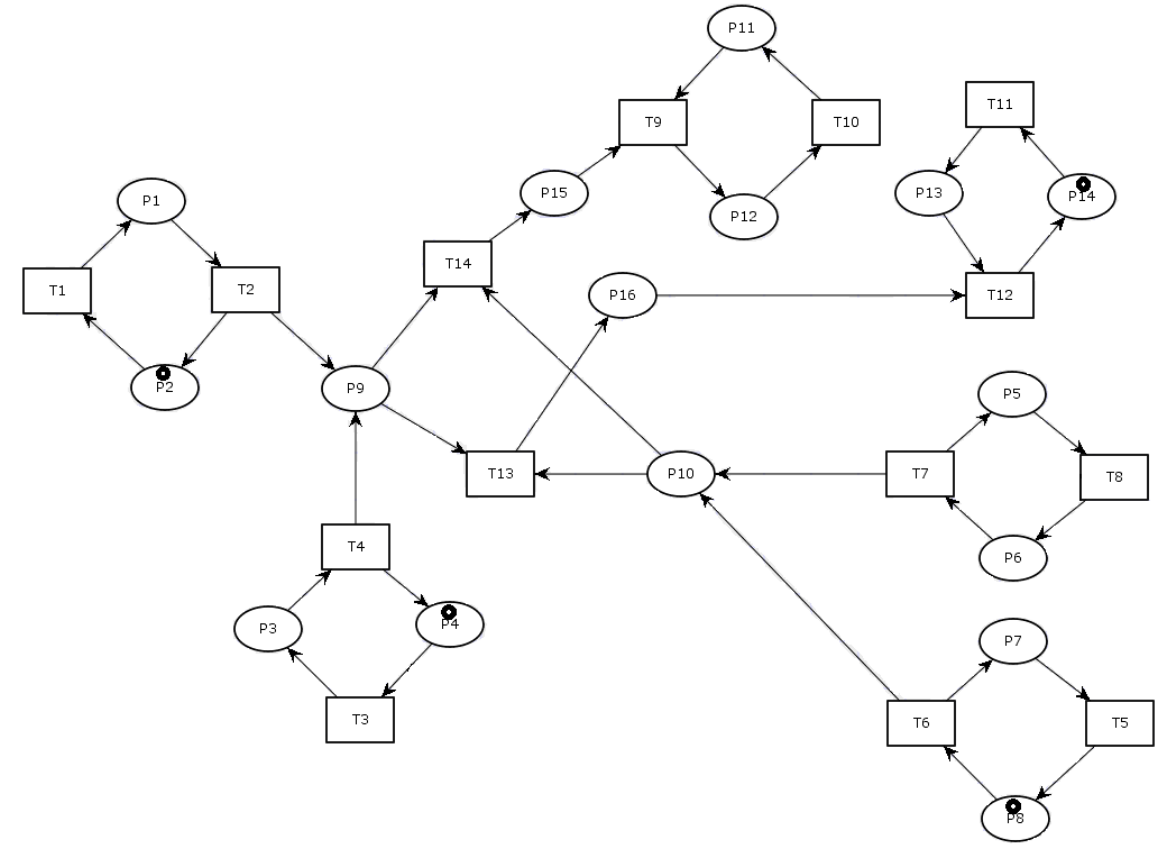

Рисунок 3.9 - Вдосконалення бази правил

- P1-P4 - генератор правил різного виду;
- P5-P8 - генератор сигналів від ЛОМ;
- P9 - вибір правила;
- P10 - вибір сигналу;
- P15, P16 - вихід сигналів різних видів;
- P11-P14 - термінатори сигналів;
- T1-T4 - генератор правил;
- T5-T8 - генератор сигналів від ЛОМ;
- T13, T14 - перевірка збігу сигналу і правила;
- T9-T12 - термінатор.

Кожен з генераторів створив по 26 сигналів, які прийшли на позиції перевірки. Там сигнали з різних позицій змогли пройти переходи. В результаті на кожну позицію прибуло по 25 сигналів (табл.3.1).

Таблиця 3.3 - Висновки

Номер позиції	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15
Всього	26	26	25	26	26	26	25	26	51	51	25	25	25	25	25

3.6 Аналіз мультиагентної системи підтримки прийняття рішень

Зазвичай тестування розглядають в якості заключної фази процесу розробки, але тестування СППР включається в кожну стадію. Розробка і тестування повинні виконуватися паралельно. За аналогією з технологією тестування традиційних програмних систем можна інтерпретувати процес верифікації (логічного тестування) як альфа-тестування програмної системи, а концептуальне тестування - як етап бета-тестування, хоча тестування СППР принципово відрізняється від тестування традиційних систем. У той час як досить суворі попередні специфікації традиційної системи дозволяють програмісту здійснювати ці роботи (особливо верифікацію системи) самостійно, для тестування СППР необхідно залучати експерта в даній галузі.

При дослідженні отриманої системи було виявлено, що при виправленні схеми, коли ОПР безпосередньо передає дані до ЛОМ, система працює швидше і точніше(рис. 3.10).

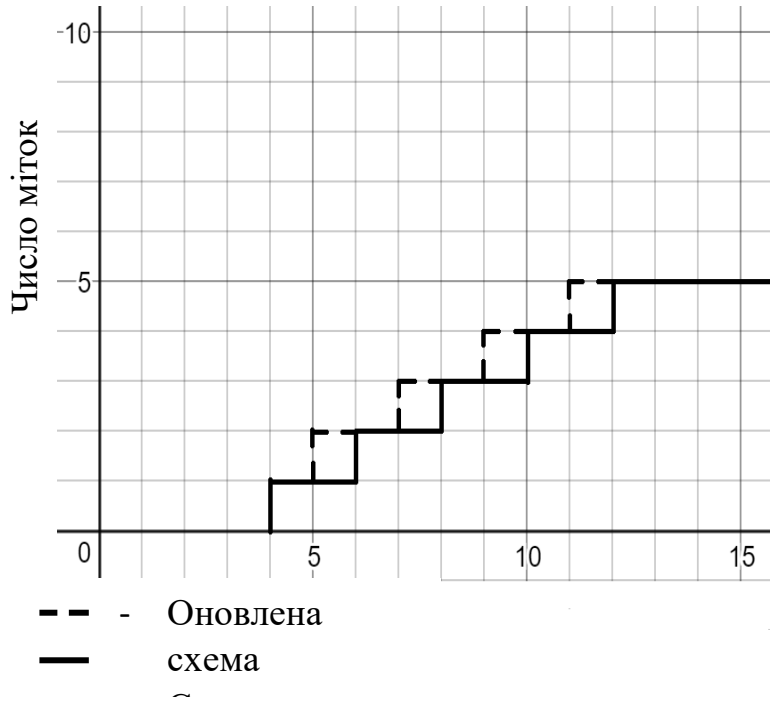

Рисунок 3.10 - Порівняння швидкості роботи системи

При впровадженні в систему генераторів і термінаторів з'явилася можливість подавати нерівномірно надходячі сигнали в необмеженій кількості(рис.3.11).

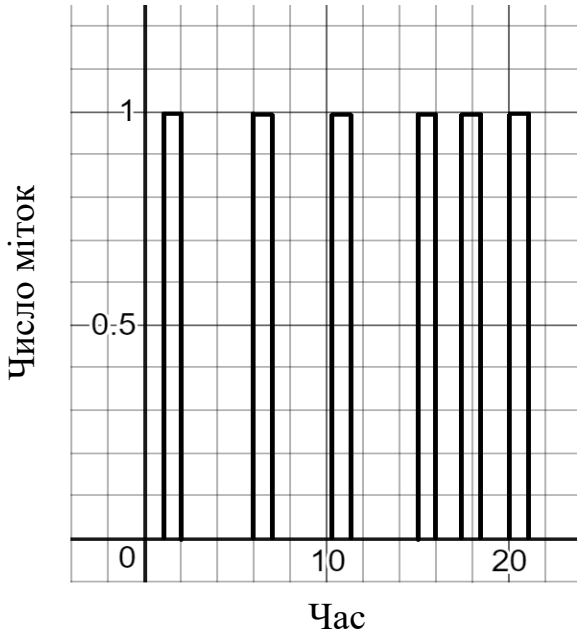

Рисунок 3.11 - Нерівномірно надходячі сигнали можуть передаватися необмежено

При роботі зі стандартною базою сигнали проходили крізь неї. При поліпшенні бази даних дані почали накопичуватися в позиції. Для виведення даних був створений перехід. Коли сигнал від ОПР приходить на перехід і одночасно з ним в базі даних є інформація, то перехід спрацьовує, а дані звіту відправляються до ОПР.

3.7 Методичні вказівки з розробки мультиагентної системи підтримки прийняття рішень в корпоративній мережі

Розробка СППР має суттєві відмінності від розробки звичайного програмного продукту. Досвід створення СППР показав, що використання при їх розробці методології, прийнятої в традиційному програмуванні, або надмірно затягує процес створення, або взагалі призводить до негативного результату.

При розробці СППР потрібно насамперед вивчити сферу застосування і призначення системи. Проаналізувати вже існуючі рішення даної проблеми. Вибрати найбільш підходящий для себе метод. У даній роботі була обрана СППР на основі методу нечіткого виводу. В ході створення СППР потрібно виділити конкретні параметри і вимоги до системи. Потрібно вирішити які функції повинна виконувати система. Дана СППР повинна аналізувати ЛВС, шукати порушення через базу правил, зберігати дані в базу даних, давати звіти на вимогу ЛПР, а також бути зручною у використанні. На основі даних функцій можна вибрати агентів, які можуть знадобитися у вашій системі.

Після цього можна вибрати контрольовані параметри і розробити базу правил.

На основі всього цього потрібно розробити структуру кожного агента і з'єднати всіх агентів в єдину систему.

Для використання мереж Петрі кожен агент представляється 2 позиціями: вхідний і вихідний.

3.8 Висновки

В ході роботи було розроблено модель мультиагентної системи підтримки прийняття рішень для роботи в корпоративній мережі.

Було проаналізовано існуючі системи підтримки прийняття рішень, що дозволило виявити основні недоліки сучасних підходів. До таких недоліків можна віднести обмеженість використання системи за кількістю людей, обмеженість роботи в рамках системи тільки з певними форматами даних, обмеженість одночасної обробки великих об’ємів інформації.

На основі структури моделі була розроблена схема, яка дозволяє моделювати ситуації, які реально відбуваються кожного дня в мережі. Створення моделі мультиагентної СППР дозволило вирішити проблему, яка пов'язана з людським фактором і надає ОПР рішення, на основі якого користувач витрачає менше часу на аналіз існуючої інформації. У даній роботі було удосконалено модель мультиагентної СППР з урахуванням взаємодії різних компонентів в мережі. Дана модель СППР здатна давати швидке рішення користувачу на основі повної та оперативної інформації і давати звіт про стан системи в цілому. Система автоматизувала пошук по базі правил і надавала ОПР рішення.

ВИСНОВКИ

В ході дослідження було розроблено мультиагентну систему підтримки прийняття рішень в корпоративній мережі.

Було проаналізовано існуючі види системи підтримки прийняття рішень, що дозволило виявити основні недоліки сучасних видів та підходів. До таких недоліків можна віднести обмеженість використання системи за кількістю людей, обмеженість роботи в рамках системи тільки з певними форматами даних, обмеженість одночасної обробки великих об’ємів інформації.

Розглянуто мультиагентий підход для розробки системи підтримки прийняття рішень при роботі в корпоративній мережі.

Проведено експеримент та виявлено особливості використання агентних технологій на прикладі системи підтримки дистанційного навчаняя. Більшість систем ДО, існуючих сьогодні в різних університетах України, майже не розглядаються комплексно як складні програмні системи. Тому цілком природним є виникнення потреби створення саме такої моделі, яка давала б цілісний погляд на процес побудови ефективних КСПДО.

Ця спроба експерементально розроблена в роботі за допомогою агентної архітектури, яка є новітньою, дуже перспективною технологією завдяки своїм інтелектуальним властивостям та можливістю оперування знаннями. Для процесу дистанційного навчання важливим є той факт, що агент здатний виконувати роль експерта, а також навчатися. Модель побудована з урахуванням основних характерних рис дистанційного навчання - гнучкість, модульність, паралельність, охват, технологічність.

Отримано такі нові наукові результати: досліджено систему дистанційного навчання як модель мультиагентної системи; розроблено модель дистанційного навчання на базі агентної архітектури JADE (Java Agent Development Framework); побудовано модель дистанційного навчання в середовищі проектування Rational

Rose Professional Edition на мові UML (Unified Modeling Language); згенеровано каркас програмного коду системи використовуючи мову програмування Java.

На основі теоретичного аналізу та проведення експерименту розроблено модель системи підтримки прийняття рішень, яка дозволяє моделювати ситуації, які реально відбуваються кожного дня в мережі. Створення моделі мультиагентної СППР дозволить вирішити проблему, яка пов'язана з людським фактором і надаватиме ОПР рішення, на основі якого користувач витрачає менше часу на аналіз існуючої інформації. У даній роботі розроблено модель роботи мультиагентної СППР з урахуванням взаємодії різних компонентів в мережі. Дана модель мультиагентної СППР дозволяє давати швидке рішення користувачу на основі повної та оперативної інформації і давати звіт про стан системи в цілому.

Також, при дослідженні отриманої системи було показано, що при виправленні схеми, коли особа, яка приймає рішення безпосередньо передає дані до локальної обчислювальної мережі, система працює швидше і точніше.

При впровадженні в систему генераторів і термінаторів з'явилася можливість подавати нерівномірно надходячі сигнали в необмеженій кількості.

При роботі зі стандартною базою сигнали проходили крізь неї. При покращенні структур бази даних дані почали накопичуватися в позиції. Для виведення даних був створений перехід. Коли сигнал від ОПР приходить на перехід і одночасно з ним в базі даних є інформація, то перехід спрацьовує, а дані звіту відправляються до ОПР. Це дозволяє зробити висновок, що дану модель можливо ще покращувати в майбутньому та адаптувати до конкретної структури мережі, з якою працює адміністратор.

ПЕРЕЛІК ПОСИЛАНЬ

1. Остін, J. L. Як робити речі зі словами / J. L. Austin.- Oxford, England: Oxford University Press, 1962.

2. Перевірка моделі для многоагентних систем: мову MABLE і його додатки / М.-Р. Хьюгет, М. Фішер, С. Парсонс // Журнал про штучний інтелект Tools.- 2006.- Vol. 15, вип. 2.-стр. 195-225.

3. Сирл, Дж. Р. Мовні акти: есе в філософії мови / J.R. Searle.- Cambridge University Press, 1969.- С. 203.

4. Ленат, Д.Д .: знання як взаємодіють експерти // Proc. Міжнародна Спільна конференція з питань штучного інтелекту. 1975.- Стор. 126-133.

5. Ленат Д. Д .: підхід штучного інтелекту до відкриття в математиці як евристичний пошук: Ph.D. дисертація / Стенфордський університет. Тисячу дев'ятсот сімдесят-шість.

6. Ага, Г. Актори: модель паралельних обчислень в розподілених системах / G. Agha.- Cambridge, MA, USA: MIT Press, 1986.- 190 стр.

7. Хьюітт, С .: Універсальний модульний формалізм ACTOR для AI // Proc. Міжнародна спільна конференція зі штучного інтелекту (IJCAI-73) .- 1973.-стор. 235-245.

8. Gasser, L. MACE: Гнучкий тест для дослідження розподілених II. / Л.Гассер, C. Braganza, N. Hermann. // Розподілений штучний інтелект. 1987.-стор. 119-152.

9. Використання ARCHON для розробки додатків DAI реального світу для транспортування електроенергії, управління і контролю прискорення частинок / N. R. Jennings, J. M. Corera, I. Laresgoiti et al. // Спеціальний випуск експерта EEE в реальному світі. Застосування систем DAI. 1996.

10. Парунак, Х. В. Д .: Основи розподіленого штучного інтелекту / Inter-science, 1994.

11. Люнберг М .: Система управління повітряним трафіком OASIS // Proc. Друга Тихоокеанська Римська міжнародна конференція по AI. Одна тисяча дев'ятсот дев'яносто дві.

12. Wooldridge, M. J. Методологія Gaia для агент-орієнтованого аналізу і проектування / M. J. Wooldridge, N. R. Jennings, D. Kinny // Автономні агенти і Multi-Agent Systems. - 2000.- Vol. 3, вип. 3.- стор. 285-312.

13. Omicini, A. SODA: суспільства і інфраструктура в аналізі і дизайні агентних систем // Proc. Перший міжнародний семінар «Агентно-орієнтованих систем»

14. Кастро, J. На шляху до розробки інформаційних систем, заснованих на вимогах: Проект тропос / Дж. Кастро, М. Колпь, Дж. Мілопулос // Інформаційні системи. 2002.- Vol. 27, вип. 6.- стор. 365-389.

15. Ю. Є. Моделювання стратегічних відносин для реінжинірингу процесів: к.т.н. Теза / Університет Торонто, факультет комп'ютерних наук. 1995.

16. Леохін Ю.Л. Архітектура сучасних систем управління корпоративними мережами. / Ю.Л. Леохін // Якість. Інновації. Освіта. - 2009. - № 2. - С. 54 - 63.

17. Лісков, О.Е. Автоматизація виробництва і переробки промислової продукції: дис. ... канд. техн. наук / О.Е. Лиско. - Орел, 2008.

18. Структура розробки Java-агентів. - Торіно, Італія, Telecom Italia Lab, 2012. - URL: http://jade.tilab.com