Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Метод проєктування програмного забезпечення на основі агентно-орієнтованої архітектури для багатокомпонентних програмних систем
Назва теми

Рівень вищої освіти Другий (магістерський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр ДРІПЗ. 240160.01.02.ПЗ

Виконав студент 2 курсу, група ІПЗм-24-1 _____ Іван ГОРБАТЮК
Підпис Ім'я, ПРІЗВИЩЕ

Керівник канд. техн. наук, доцент _____ Юрій ФОРКУН
Науковий ступінь, звання Підпис Ім'я, ПРІЗВИЩЕ

Нормоконтролер канд. техн. наук, доцент _____ Оксана ЯШИНА
Підпис Ім'я, ПРІЗВИЩЕ

До захисту допускаю:
Завідувач кафедри
інженерії програмного забезпечення _____ Леонід БЕДРАТЮК
Підпис Ім'я, ПРІЗВИЩЕ

Дата
15 грудня 2025

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Другий (магістерський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІПЗ
Л. П. Бедратюк
01.09.2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Горбатюку Івану Володимировичу
Прізвище, ім'я, по батькові здобувача

1. Тема роботи: Метод проєктування програмного забезпечення на основі агентно-орієнтованої архітектури для багатокомпонентних програмних систем

Керівник роботи канд. техн. наук, доцент Форкун Ю.В.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 25.08.2025 р. № 65
2. Строк подання студентом роботи на кафедру 01.12.2025 р.
3. Вихідні дані до роботи Матеріали науково-дослідної практики, методичні вказівки до виконання кваліфікаційної роботи
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз предметної області і виявлення наявних проблем та завдань
2. Підходи та методи вирішення поставлених задач
3. Проектування програмного забезпечення
4. Реалізація та тестування програмного забезпечення
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Презентаційні матеріали (слайди)

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Антиплагіат	Ю. В. Форкун	1.12.25	15.12.2025
Нормоконтроль	О. М. Яшина		

7. Дата видачі завдання « 02 » вересня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1 Вивчення предметної області, формування мети та задач дослідження, визначення об'єкта та предмета дослідження	29.10-26.10.2025	виконано
2 Робота над розділом 1 – вивчення джерел, аналіз сучасних підходів, методів і засобів за темою роботи, визначення задач, висновки	29.10-26.10.2025	виконано
3 Робота над розділом 2 – розробка підходів дослідження поставленої задачі, висновки	27.10-02.11.2025	виконано
4 Робота над науковими статтями	03.11-09.11.2025	виконано
5 Робота над розділом 3 – аналіз вимог до програмного застосунку, розробка структури, вибір технологій, висновки	10.11-16.11.2025	виконано
6 Робота над розділом 4 – реалізація та тестування програмного забезпечення	17.11-23.11.2025	виконано
7 Попередній захист дипломної роботи	24.11-31.11.2025	виконано
8 Узгодження постановки задач, отриманих результатів та висновків, написання вступу, загальних висновків, оформлення джерел та додатків, оформлення записки та графічних матеріалів відповідно до вимог	01.12-07.12.2025	виконано
9 Перевірка роботи на наявність плагіату, нормоконтроль, брошурування пояснювальної записки, підготовка супровідних документів	08.12-14.12.2025	виконано
10 Підготовка до захисту кваліфікаційної роботи	з 15.12.2025 р.	виконано

Студент _____ Горбатюк І.В.
Підпис Ім'я, ПРІЗВИЩЕ

Керівник роботи _____ Форкун Ю. В.
Підпис Ім'я, ПРІЗВИЩЕ

РЕФЕРАТ

Тема кваліфікаційної роботи: Метод проєктування програмного забезпечення на основі агентно-орієнтованої архітектури для багатокомпонентних програмних систем.

Автор роботи: Горбатюк Іван Володимирович

Керівник роботи: канд. техн. наук, доцент Форкун Юрій Вікторович

Пояснювальна записка 104 с., 33 рис., 4 дод., 40 джерел.

Ключові слова: АГЕНТНО-ОРІЄНТОВАНА АРХІТЕКТУРА, МУЛЬТИАГЕНТНА СИСТЕМА, МЕТА-МОДЕЛЬ, ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, МЕТА-АНАЛІЗ.

Об’єктом даного дослідження є агентно-орієнтована архітектура, особливості її реалізації та способи роботи у багатокомпонентних програмних системах.

Предметом дослідження виступають методи проєктування багатокомпонентних програмних ситем на основі агентно-орієнтованої архітектури для багатокомпонентних програмних систем їх роль у формуванні рішень та можливість комбінування у єдину систему.

У дослідженні вперше запропоновано метод проектування агентів-приймачів рішень, кожен з яких відповідає за оцінку окремого аспекту моделі. Вперше поєднано характеристики агентів, застосунку та мета-моделей, що дозволяє здійснювати вибір моделі поведінки агентів-приймачів в загальній архітектурі системи.

Метою роботи є дослідження, аналіз та розробка методу щодо автоматизованого вибору мета-моделей при побудови рішень для проєктування програмного забезпечення на основі мультиагентних систем агентно-орієнтованої архітектури, яка має враховувати характеристики предметної області, особливості моделей агентів та вимоги до самого застосунку.

Для досягнення поставленої мети та завдання нами було проведено детальний аналіз предметної області, проаналізовано сучасні дослідження. На основі чого було

розроблено метод та механізм і протестовано при розробці архітектури мультиагентної системи.

Наукова новизна роботи полягає в перше запропонованому методі проєктування ПЗ на основі агентно-орієнтованої архітектури для багатокомпонентних програмних систем, який базується на агентах-приймачах рішень, де кожен агент оцінює окремий аспект моделі. Уперше поєднано характеристики, агентів, специфікації задачі, застосування мета-моделей що забезпечує обґрунтований вибір поведінки агентів і варіанту архітектурної інтеграції в загальній структурі системи.

Результатом є розроблений метод та механізм проектування взаємодії агентів в багатокомпонентних системах, які забезпечує системне, обґрунтоване та гнучке визначення найкращої мета-моделі серед багатьох альтернатив.

Практичне значення полягає в тому, що запропонований метод і механізм узгодження зменшують початкову невизначеність вибору методології та моделі, підтримують прийняття рішень на ранній стадії та знижують ризики помилок у подальшій розробці. Реалізований прототип демонструє можливість використання підходу як інструмента підтримки проєктувальника під час побудови багатокомпонентних агентних систем.

З теоретичних методів дослідження було обрано: аналіз, синтез, порівняння та моделювання.

Емпіричними методами нашого дослідження виступили: опис, тестування.

05.12.25
Дата

Підпис

ABSTRACT

Qualification thesis: Method of software design based on agent-oriented architecture for multicomponent software systems.

The author of the work: Gorbatyuk Ivan Vladimirovich.

Supervisor: Cand. Tech. Sci., Associate Professor Forkun Yuriy Viktorovych

Explanatory note 104 p., 33 figs., 3 additions, 40sources.

Keywords: AGENT-ORIENTED ARCHITECTURE, MULTI-AGENT SYSTEM, META-MODEL, , SOFTWARE ENGINEERING, META-ANALYSIS.

The object of this study is agent-oriented architecture, its features of its implementation and ways of working in multicomponent software systems.

The subject of the study is the methods of software design based on agent-oriented architecture for multicomponent software systems, their role in the formation of solutions and the possibility of combining into a single system.

In the study, for the first time, a method for designing decision-making agents is proposed, each of which is responsible for evaluating a different aspect of the model. For the first time, the characteristics of agents, application and meta-models are combined, which allows you to select a model of the behavior of receiving agents in the general architecture of the system.

The purpose of the work is to research, analyze and develop a method for the automated selection of meta-models when building software design solutions based on multi-agent systems of agent-oriented architecture, which should take into account the characteristics of the subject area, the features of agent models and the requirements for the application itself.

To achieve the set goal and task, we conducted a detailed analysis of the subject area, analyzed modern research. On the basis of this, the method and mechanism were developed and tested during the development of the multi-agent system architecture.

The scientific novelty of this work lies in the first-time proposed method for software design based on an agent-oriented architecture for multi-component software systems, which is founded on decision-making agents, where each agent evaluates a specific aspect of the model. For the first time, the characteristics of agents, task specifications, and the use of meta-models are integrated, enabling a well-founded selection of agent behavior and an appropriate architectural integration variant within the overall system structure.

The result is a developed method and mechanism for designing agent interactions in multicomponent systems, which provides a systematic, reasonable, and flexible definition of the best meta-model among many alternatives.

The practical significance of the work lies in the fact that the proposed method and matching mechanism reduce the initial uncertainty in the selection of a methodology and model, support decision-making at an early stage, and decrease the risk of errors in subsequent development. The implemented prototype demonstrates the feasibility of using the proposed approach as a designer support tool in the development of multi-component agent-based systems.

Of the theoretical research methods, the following were chosen: analysis, synthesis, comparison and modeling.

The empirical methods of our research were: description, testing.

ЗМІСТ

ВСТУП

Сучасним програмним системам притаманна значна схильність до складності. Популярність розподілених і багатокомпонентних складних систем – як таких, що працюють на мобільних пристроях, стаціонарних системах, які працюють будь-де в організаціях, чи в області домашньої автоматизації, вимагає зусилля розробки систем, які виходять за межі традиційних методів розробки такого програмного забезпечення.

Таке широке та досить різноманіття варіантів застосування ускладнює початковий вибір, як для системного розробника так і проектувальника. Іншими словами, проектувальник та розробник повинні вибрати з багатьох можливих варіантів та компонентів відповідну до предметної області адекватну методологію для цілей створюваної ним системи. Такий підхід викликає початкову невизначеність у проектувальників та розробників. А це в свою чергу призводить до використання БПС, як життєздатного варіанту для проектування та розробки відповідних проектних рішень.

Зокрема, до зараз, багато схожих архітектур проектування ПЗ є досить складними та важкими для використання з досить різних причин, а їх невірна реалізація може, зокрема, призвести до таких наслідків, які навіть зможуть унеможливити подальшу розробку застосунку та програмного засобу.

Виходячи з наведених вище фактів їх використання, ми вважаємо, що їм бракує певної підтримуючої альтернативи, яка б могла об’єднати ці дві частини. Така альтернатива повинна мати можлиість підтримувати вибір задачі адекватної специфікації компонентів багатокомпонентних програмних систем, що дозволить побудувати програмний застосунок з урахуванням проблемної предметної області.

У нашій роботі ми пропонуємо покрити цей недолік за допомогою механізму, який гадає можливість проектувальнику у процесі вибору методології, пропонуючи йому найбільш підходящі моделі та, відповідно, найкращий метод архітектурної інтеграції для досягнення його цілей рішення.

Створення такого механізму ще більше зробить більш досяжним використання методологій на основі багатокомпонентних програмних систем для розробників, а також сприятиме широкому використанню багатокомпонентних програмних систем, ще більше розширюючи та даючи можливості для задуму та проектування та розробки відповідного програмного забезпечення.

Таким чином, об’єктом даного дослідження є агентно-орієнтована архітектура, її особливості її реалізації та способи роботи у багатокомпонентних програмних системах.

Предметом дослідження виступають методи проєктування багатокомпонентних програмних систем на основі агентно-орієнтованої архітектури для багатокомпонентних програмних систем їх роль у формуванні рішень та можливість комбінування у єдину систему.

У дослідженні вперше запропоновано метод проектування програмного забезпечення на основі агентно-орієнтованої архітектури для багатокомпонентних програмних систем на основі агентів-приймачів рішень, кожен з яких відповідає за оцінку окремого аспекту моделі. Вперше поєднано характеристики агентів, застосунку та мета-моделей, що дозволяє здійснювати вибір моделі поведінки агентів-приймачів в загальній архітектурі програмної системи.

Метою роботи є дослідження, аналіз та розробка методу щодо автоматизованого вибору мета-моделей при побудови рішень для проєктування програмного забезпечення на основі багатокомпонентних програмних систем агентно-орієнтованої архітектури, яка має враховувати характеристики предметної області, особливості моделей агентів та вимоги до самого застосунку.

Для досягнення поставленої мети та завдання нами було проведено детальний аналіз предметної області, проаналізовано сучасні дослідження, проаналізовані сучасні методи взаємодії агенті між собою. На основі чого було розроблено метод та механізм і протестовано при розробці архітектури мультиагентної багатокомпонентної системи.

Для досягнення поставлених цілей нашої роботи, вважаючи вище викладене, нами було прийнято рішення про розробку методу вибору архітектури проєктування

програмного забезпечення на основі агентно-орієнтованої архітектури для багатокомпонентних програмних систем, яке б дозволило покращити початкову ситуацію з вибором відповідної архітектури, з системою правил, яка допоможе приймати рішення на початковому етапі та в результаті щвидко та коректно здійснювати розробку таких програмних продуктів.

Крім того у кваліфікаційній роботі розроблено структуру програмного забезпечення на основі багатокомпонентних програмних систем агентно-орієнтованої архітектури. В результаті було створено програмний застосунок і проведено тестування та валідацію програмного застосунку. Для цього було проведено ряд тестів для створеного програмного забезпечення, з допомогою яких було здійснено аналіз отриманих результатів та показано ефективність розробленого методу і зроблені висновки щодо можливостей його подальшого застосування та використання.

Таким чином наша робота дозволяє підвищити рівень проєктування програмного забезпечення на основі агентно-орієнтованої архітектури для багатокомпонентних програмних систем в галузі інженерії програмного забезпечення, надаючи механізм для оптимізації вибору архітектурних рішень допомагаючи приймати початкові рішення та зменшуючи ризик в подальшій розробці програмного забезпечення.

Робота надає можливість для нових перспективних розробок та можливість подальших досліджень у застосуванні архітектурних агентно-орієнтованих рішень в багатокомпонентних системах, розширюючи таким чином їх можливості застосування для коректної розробки програмного забезпечення.

Наукова новизна роботи полягає в перше запропонованому методі проєктування ПЗ на основі агентно-орієнтованої архітектури для багатокомпонентних програмних систем, який базується на агентах-приймачах рішень, де кожен агент оцінює окремий аспект моделі. Уперше поєднано характеристики, агентів, специфікації задачі, застосування мета-моделей що забезпечує обґрунтований вибір поведінки агентів і варіанту архітектурної інтеграції в загальній структурі системи.

Результатом є розроблений метод та механізм проектування взаємодії агентів в багатокомпонентних системах, які забезпечує системне, обґрунтоване та гнучке визначення найкращої мета-моделі серед багатьох альтернатив.

Практичне значення полягає в тому, що запропонований метод і механізм узгодження зменшують початкову невизначеність вибору методології та моделі, підтримують прийняття рішень на ранній стадії та знижують ризики помилок у подальшій розробці. Реалізований прототип демонструє можливість використання підходу як інструмента підтримки проєктувальника під час побудови багатокомпонентних агентних систем.На основі здійсненого дослідження у роботі нами було опубліковано тези у Збірнику наукових праць конференції АПКН-2025 де висвітлено основні питання результати досліджень з питань проєктування програмного забезпечення на основі агентно-орієнтованої архітектури для багатокомпонентних програмних систем в галузі програмної інженерії.

1 ХАРАКТЕРИСТИКА ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області та і виявлення наявних проблем та завдань

Багатокомпонентні складних системи (БПС), загалом, це складні програмні або апаратно-програмні утворення, що складаються з множини автономних агентів, кожен з яких має власні цілі, ресурси, знання та стратегії поведінки. Агент у такій системі здатен самостійно сприймати середовище, приймати рішення та взаємодіяти з іншими агентами, що робить Багатокомпонентні складні системи особливо корисними у ситуаціях, де централізоване керування є неефективним або неможливим. У цих системах загальна поведінка не задається жорсткими алгоритмами, а є результатом колективної взаємодії агентів, які координують дії, співпрацюють, змагаються або формують коаліції для досягнення більш складних цілей. Така децентралізована організація забезпечує високу гнучкість, масштабованість і стійкість до відмов, оскільки втрата одного агента рідко призводить до зупинки всієї системи. Мультиагентні підходи широко використовуються в робототехніці, логістиці, моделюванні соціальних процесів, управлінні транспортними мережами, фінансових симуляціях та інтелектуальних системах підтримки прийняття рішень.

Другим важливим аспектом багатокомпонентних складних систем є їх здатність до адаптації та самоорганізації, що дозволяє системам ефективно працювати в динамічних, частково невизначених або непередбачуваних середовищах. Агенти можуть коригувати власну поведінку, враховуючи зміну умов або дії інших суб’єктів, що забезпечує виникнення колективного інтелекту, подібного до поведінки біологічних систем, наприклад мурашиних колоній або зграї птахів. Багатокомпонентні складні системи можуть моделювати поведінку великої кількості незалежних учасників, відтворюючи складні emergent-ефекти, які неможливо описати у рамках традиційних централізованих моделей. Це робить мультиагентні системи потужним інструментом для прогнозування, оптимізації,

розподіленого керування та створення інтелектуальних систем нового покоління, здатних працювати у масштабних і складноструктурованих доменах.

Прикладом такої багатокомпонентної складної системи та одноагентної системи нами наведено на рисунку 1.1.1

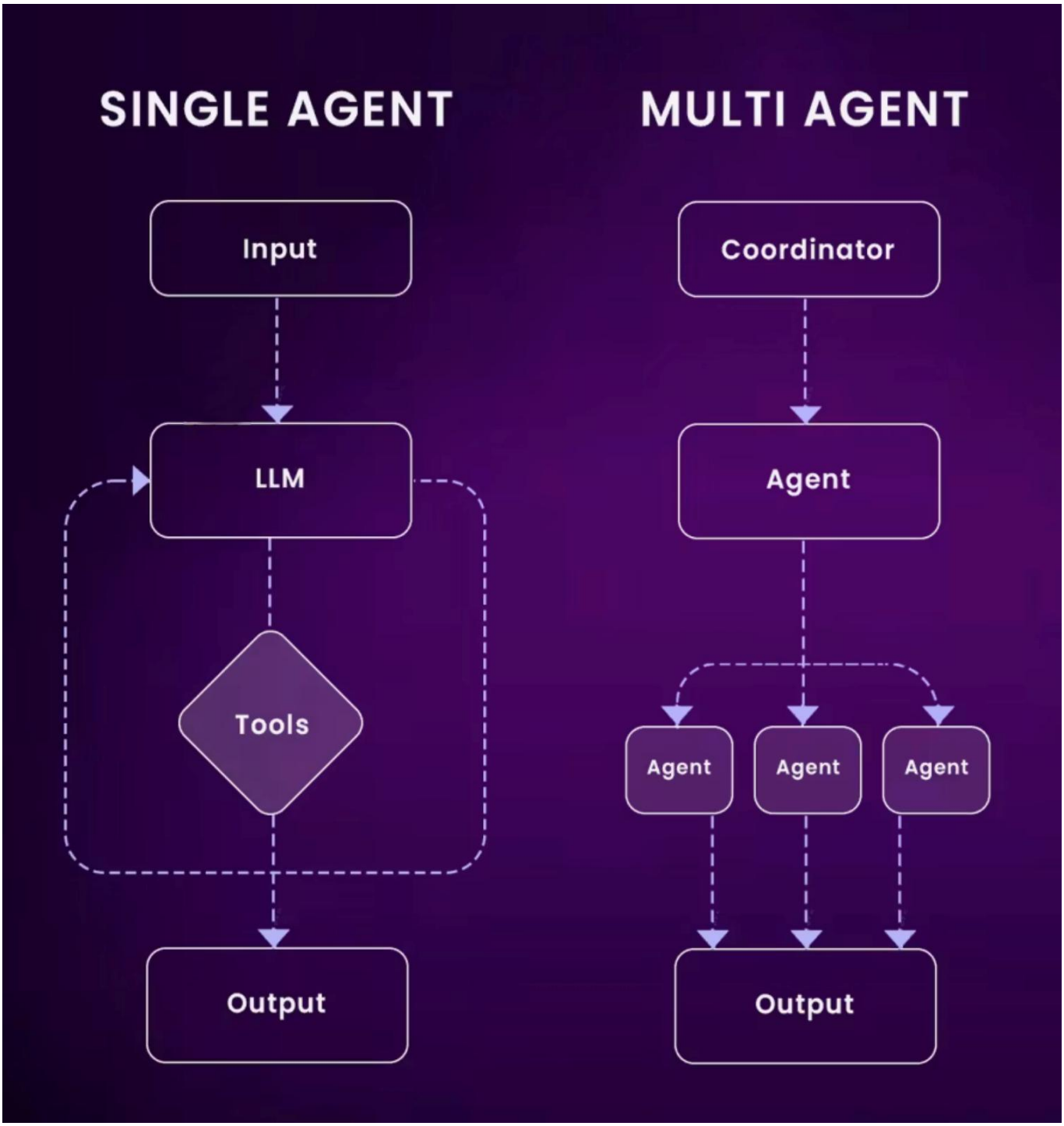


Рисунок 1.1.1 – Одноагетні та багатокомпонентні системи

Архітектурний тип багатокомпонентних складних систем є однією з парадигм, яка використовуються саме для заповнення такої прогалини. Парадигма БПС дозволяє моделювати систему за допомогою підходів, що дозволяють через їх потенціал переосмислити взаємодію машин та людей і вирішувати складні проблеми, не тільки взаємодії самих механізмів, а інші задачі в різних організаціях та установах. До теперішнього часу було проведена значна кількість наукових досліджень та було докладено багато зусиль та для спрощення розробки таких БПС. Однак, до цього часу, більша частина цих досліджень виконувалася без координації і відповідно в різних напрямках. Крім того, широке різноманіття різних рішень ускладнює вибір уже інснуючих моделей.

До головних переваг багатокомпонентних складних систем зазвичай відносять кілька ключових характеристик, що роблять їх особливо ефективними для складних, динамічних і розподілених задач.

Серед них можна назвати децентралізацію та відсутність єдиного центру керування. БПС не потребують центрального контролера – кожен агент приймає рішення автономно. Це, в свою чергу забезпечує більшу гнучкість, відсутність критичної точки відмови, природну масштабованість системи.

Зокрема масштабованість у системі дає можливість легко додавати нових агентів без істотної перебудови архітектури. Зі збільшенням кількості агентів продуктивність часто зростає, а не зменшується, що робить БПС ефективними для великих доменів.

Багатокомпонентні складні системи також володіють стійкістю до відмов (fault tolerance). Через децентралізовану природу вони можуть працювати навіть тоді, коли окремі агенти виходять з ладу. Система адаптується і перебудовує взаємодію без повного зупинення.

Вони також мають відмінну адаптивність та можливість самоорганізації. Агенти здатні змінювати поведінку залежно від стану середовища, дій інших агентів, появи нових задач тощо. Це робить багатокомпонентні складні системи дуже ефективними у непередбачуваних або швидкозмінних умовах.

До основних плюсів цих систем також слід віднести, природне моделювання складних систем. Багато реальних процесів є багатосуб'єктними за природою , такі як рух транспорту, поведінка ринку, зграї тварин, логістика тощо. БПС дозволяють моделювати ці явища більш точно, ніж традиційні централізовані моделі.

Багатокомпонентні складні системи володіють високою продуктивністю завдяки розподіленості. Агенти можуть працювати паралельно, що підвищує загальну швидкість обробки інформації та виконання задач.

Гнучкість у розв'язанні задач для багатокомпонентних складних систем полягає в тому, що кожен агент може мати власні алгоритми, стратегії та ресурси.

Це дозволяє легко комбінувати різні підходи, додавати нові функціональні можливості та адаптувати систему до різних умов.

Основні недоліки багатокомпонентних складних систем, які зазвичай виділяють у науковій літературі та практичній розробці пов'язані з високою складністю проєктування, координації та забезпечення передбачуваної поведінки агентів.

Зокрема, складність проєктування та моделювання полягає в тому, що вони мають децентралізовану архітектуру, де рішення приймаються локально кожним агентом. Через це важко передбачити глобальну поведінку системи, моделювання взаємодій між агентами стає складним, необхідно використовувати спеціальні методології (Gaia, ADELFE, MaSE тощо).

Висока обчислювальна і комунікаційна складність багатокомпонентних складних систем полягає в тому, що оскільки агенти постійно взаємодіють то зростає обсяг комунікацій, збільшується навантаження на мережу, можуть виникати проблеми з синхронізацією.

У великих багатокомпонентних складних систем це стає особливо критичним. Непередбачуваність emergent-поведінки, що виникає зі взаємодії агентів, може бути корисною, але часто — непередбачуваною або небажаною. А це ускладнює тестування, верифікацію та гарантування безпеки системи.

Проблеми з координацією та узгодженням дій у децентралізованих системах важко забезпечити через стабільну координацію, розв'язання конфліктів інтересів,

узгоджені рішення між великою кількістю агентів. Іноді необхідні додаткові протоколи або механізми соціальної поведінки.

У таких системах є складність тестування та верифікації, їх складно повністю протестувати, оскільки кількість можливих сценаріїв взаємодії дуже велика, поведінка динамічна і залежить від локальних рішень агентів, необхідна симуляція багатьох станів середовища.

У цих систем підвищені вимоги до ресурсів, через паралельність і автономність агентів системи можуть потребувати більше пам’яті, більше обчислювальних ресурсів, додаткові механізми моніторингу.

Також у багатокомпонентних складних систем є складність реалізації безпекових механізмів. У розподіленій системі важче забезпечити перевірку прав доступу, захист від несанкціонованих дій, контроль помилкової або некоректної поведінки агента.

Проте, головна проблема полягає у тому, що для будь-якого проектувальника перед початком створення певного проекту, є певний набір опцій та задача, яку потрібно певним чином вирішити.

Розробник повинен узгоджувати характеристики своєї задачі з компонентами, які належать до одного, або декількох методологічних підходів, які можуть реалізувати шлях до побудови рішення цієї задачі. Таким чином, рівень досвіду і попередні знання вже наявних методологій відіграють ключову роль у прийнятті початкових рішень. методологій відіграють ключову роль у прийнятті початкових рішень.

Наприклад, такі методологія ADELFE, це - спеціалізований підхід до аналізу, проектування та, розробки багатокомпонентних програмних систем, орієнтовані на автономні, кооперативні та адаптивні агенти, працюють у різних середовищах. Цю методологію створено у рамках європейського проєкту ADELFE (Agent Development Environment for Flexible Engineering) , і вона базується на підході AMAS – Adaptive Multi-Agent Systems.

Проте дана методологія орієнтована лише на сферу багатокомпонентних програмних систем, так само як методологія Gaia, яка допомагає розробникам,

проаналізувати предметну область, визначити ролі агентів, описати взаємодію між агентами та побудувати архітектуру багатоагентної програмної системи. Gaia перетворює складну систему зі взаємодіючих агентів у зрозумілу структуру з чітко визначеними ролями та протоколами.

Також вона спрямовані лише на закриту область зі статичними характеристиками. Тобто, проектувальник повинен знати обидві сторони перед вибором маршруту.

Підсумовуючи, можна виявити, що загалом визначається як система, яка складається з численних інтелектуальних агентів, які взаємодіють разом для спільного виконання завдання. Також зазначено, що для визначення парадигми мультиагентності необхідно враховувати різні точки зору.

Перша і головна точка зору – це інженерія програмного забезпечення. БПС парадигма розглядається як підхід, що дозволяє проектувати автономне програмне забезпечення, здатне взаємодіяти з різними частинами розподіленої системи (Рисунок 1.1.2). Зазвичай здатність взаємодії розглядається як найважливіша характеристика, що забезпечується таким підходом. Крім того, ця парадигма дозволяє проектувати архітектури для створення програмних рішень складних штучних систем.

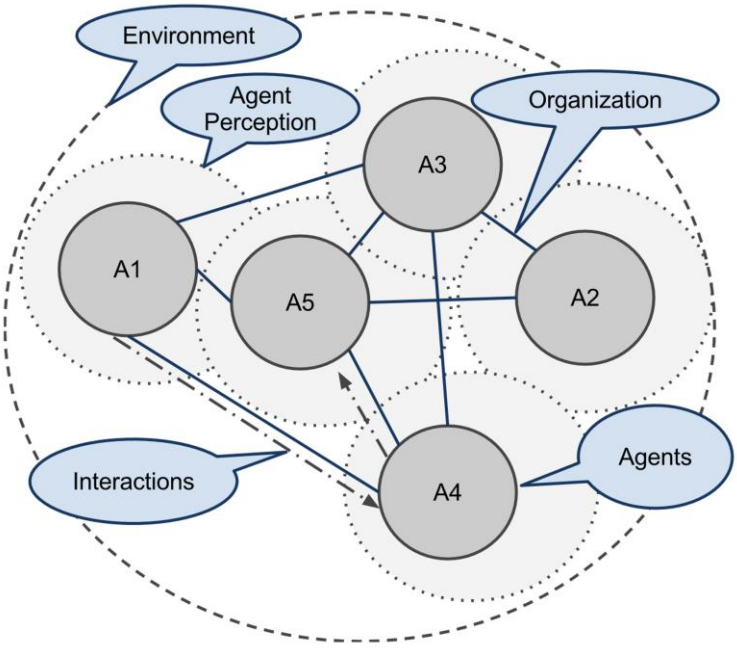


Рисунок 1.1.2 – Багатоагентна програмна система

З огляду на підхід розподілених систем, друга точка зору, це парадигма багатьох агентів, яка забезпечує підхід, з якого можна уявити розподілені рішення з використанням самих агентів.

І третя точку зору, що БПС є інструментом людської спільноти. З цієї точки зору, як взаємодії людини в людських суспільстві парадигма, БПМ абстрагує їх індивідуальні особливості, спеціалізацію, організацію та взаємодію. Це також дозволяє розробити симуляцію нашого соціуму, який заснований на людині, як основі системи.

Таким чином, на основі вище викладеного можнемо дійти висновку, що багатоагентна програмна система є досить затребуваною, проте складна в процесі як її вибору та проектування і реалізації. На її коректну роботу, її ефективність впливають дуже велика кількість факторів та умов.

1.2 Аналіз сучасних досліджень та рішень

Існуюче різноманіття методів проектування архітектури для багатокомпонентних програмних систем, які орієнтовані на інженерію ПЗ заданих проблем в конкретних областях, ускладнює розробникам вибір правильного підходу. Це, зокрема, означає, що коли береться певна задача, її відразу намагаються вирішити за допомогою деякого якого мультиагентного підходу чи методу. Зазвичай, в основному це робиться без якогось попереднього аналізу, якщо характер задачі чи проблеми збігається з предметною областю рішення, що охоплюєтьсяцтм обраним методом. Таким чином, наша основна тезна проблема полягає в тому, що існує багато методологій, багато проблем і безліч можливих рішень для БПС. Тому рішення про розв'язання проблеми за допомогою рішення на основі БПС не є простим. Також, враховуючи, що у нас низький досвід розробки БПС, це ще більше ускладнюється, а ймовірність збою при використанні БПС стає більшою.

Проблема, що розглядається в цьому проекті, полягає в необхідності загального механізму попереднього аналізу, здатного уніфікувати вибір специфікації моделей в залежності від проблемної області і цільового застосування. Дана робота зосереджена на одній головній проблемі, пов'язаній з необхідністю попереднього етапу аналізу перед вибором БПС методології для розгортання системи. Цю основну проблему можна пояснити з двох різних підходів.

Перший підхід - це характеристика проблеми, яка якраз і знаходиться в аналізі опису проблеми. Другий підхід знаходиться в компонентах або моделях, які використовуються для створення БПС системи.

Мислення в єдиному глобальному підході до розробки програмного забезпечення для створення БПС систем неможливою, тому що домени та проблеми дуже неоднорідні. Однак, розглядаючи ті проблеми, які зазвичай ефективно вирішуються за допомогою БПС, це:

– моделювання БПС;

– виправлення неполадок ;

– інтеграція програмного забезпечення, системи управління та інших наведених вище систем.

Ми вважаємо, що проста модель створення БПС для всіх них неможлива. Зазвичай ці проблеми вирішуються за допомогою підходів, цілеспрямованих і спеціалізованих. Наприклад, як ми можемо уніфікувати дизайн додатків як віддалених, таких як Інтернет, веб-служби або виробничі системи?

Парадигма багатоагентності успадковує риси від різних теорій прийняття рішень і соціальних. Кожна агентна сутність може бути утворена різними типами здібностей від однієї реактивної до висококогнітивної. Такі сутності інтегруються в різні архітектури операціоналізують моделі агентів, навички взаємодії, сприйняття та обробки завдань у середовищі.

З соціальної точки зору багатоагентні ознаки інтегровані в різні способи взаємодії та організаційні структури, пов'язані з природою агентів. Цей проект також зосереджується на проблемі неузагальнюваної моделі, пропонуючи рішення в характеристиці мета-моделей на тому ж попередньому етапі аналізу.

Більш детальну інформацію про методи проектування багатокомпонентних програмних систем на основі агентів та агентно-орієнтованого підходу наведено в таблиці 1.2.1

Таблиця 1.2.1 – Таблиця класифікації методів проєктування багатокомпонентних програмних систем

№	Метод (Категорія)	Класифікація	Приклади методів та реалізацій	Опис
1	Організаційно-орієнтовані методи	Розглядають MAS як організацію, де агенти виконують ролі, підпорядковуються нормам і взаємодіють через протоколи. Фокус - структура, ролі, взаємодія.	Gaia, MOISE+, OperA	Фокус на ролях, протоколах, організаційній структурі.
2	Адаптивні та кооперативні методи	Агенти мають вроджені механізми кооперації, адаптації та реакції на конфліктні ситуації. Система самоорганізується без централізованого контролю.	ADELFE, AMAS-підхід, методи самокоординації	Фокус на адаптивності та кооперації.
3	Вимогово-орієнтовані (goal-орієнтовані методи)	Проектування MAS через цілі, підцілі та залежності між учасниками. Фокус на моделях мотивації, цілях і залежностях між агентами.	Tropos, i* Framework	Фокус на цілях та залежностях між акторів/агентами.
4	Інженерно-процесні (повний життєвий цикл орієнтовані методи)	Дається повний цикл розробки: аналіз, проєктування, реалізація, тестування для агентних систем. Методології схожі на класичні SE-процеси, але з акцентом на агентність.	MaSE, O-MaSE, SODA	Фокус на цілях та залежностях між акторів/агентами.
5	Онтологічні та знання-орієнтовані методи	Проєктування MAS як системи, що оперує знаннями, онтологіями та семантикою. Фокус на спільному розумінні домену й обміні знаннями між агентами.	MAS-CommonKADS, OntoAIMS	Фокус на формальному описі знань і онтологій.
6	Соціально-відповідальні та нормативні методи	Агенти розглядаються як учасники соціальної системи з правилами, зобов’язаннями та	OperA, Normative MAS, ISLANDER	Фокус на соціальних правилах, зобов’язаннях і нормах, які визначають

		договорами.		поведінку агентів у спільному середовищі.
7	BDI-моделі (Belief–Desire–Intention)	Агенти проєктуються як раціональні суб’єкти з віруваннями, бажаннями та намірами.	BDI-модель, Jadex, PRS	Фокус на раціональній поведінці агентів, змодельованій через їхні вірування, бажання та наміри.
8	Інженерія самоорганізованих систем	Орієнтація на emergent-поведінку, локальні правила та принципи колективного інтелекту (як у зграях або колоніях).	Методи swarm intelligence, Ant Colony Systems	Фокус на локальних простих правилах, що породжують складну емерджентну поведінку без централізованого керування.
9	Формальні та математичні методи	Використання формальних моделей (логіки, теорії ігор, автоматів) для строгого аналізу та верифікації поведінки агентів.	Temporal Logic MAS, Game-theoretic MAS	Фокус на строгому математичному описі та верифікації поведінки агентів і системи загалом.
10	Моделювання агентів у симуляційних середовищах	озробка МАС через моделювання великих популяцій агентів з акцентом на поведінку та емерджентні ефекти.	**Repast**, NetLogo, AnyLogic (agent-based mode)	Фокус на швидкому прототипуванні та експериментальному аналізі поведінки великих популяцій агентів.

Розглянемо кожний метод наведений в таблиці більш детальніше для вирішення постановки завдання нашого дослідження та вибору технологій вирішення задачі.

Організаційно-орієнтовані методи розглядають багатоагентну систему як штучну організацію, у якій кожен агент виконує певну роль відповідно до визначених правил і структур. Основна увага приділяється моделюванню ролей, протоколів взаємодії, відповідальностей і повноважень, що дозволяє створювати системи з передбачуваною колективною поведінкою. Такі методи, як Gaia і MOISE+, забезпечують чітке структурне проєктування складних систем із багатьма взаємодіючими агентами. Попри це, вони менш ефективні у динамічних умовах, де структура взаємодій швидко змінюється.

Адаптивні та кооперативні методи засновані на концепції самоорганізації та кооперативної поведінки, коли агенти автономно реагують на зміни середовища та взаємодій. У центрі уваги здатність агентів виявляти та вирішувати некооперативні ситуації, що сприяє підвищенню стійкості й адаптивності системи. Методологія ADELFE та AMAS-підхід демонструють, як локальні правила можуть породжувати складну, але узгоджену колективну поведінку. Основний недолік такого підходу полягає у труднощах верифікації emergent-поведінки та обмеженій передбачуваності глобальних результатів.

Goal-oriented підходи зосереджені на моделюванні системи через ієрархію цілей, підцілей, залежностей між ними та ролей учасників. Методології Tropos і i* Framework дозволяють детально описати мотивацію агентів, їхні взаємні зобов'язання та соціальні залежності. Це робить їх особливо корисними на ранніх етапах проєктування, де важливо зрозуміти логіку цілей. Проблемою є складність переходу від високорівневих моделей цілей до конкретної архітектури та поведінки агентів.

Процесно-орієнтовані методи пропонують повноцінний інженерний цикл розробки БПС від аналізу до тестування. MaSE та O-MaSE забезпечують структуровані кроки побудови агентів, визначення їхніх ролей, поведінок і взаємодій. Цей підхід дозволяє систематизувати розробку та полегшує інтеграцію у традиційні інженерні процеси. Проте формальність і багатоступеневість роблять методологію досить важкою для малих або швидких проєктів.

Онтологічні підходи ґрунтуються на поданні знань, якими агенти обмінюються або користуються під час виконання завдань. Методи, такі як MAS-CommonKADS, дозволяють формалізувати знання, що забезпечує точну комунікацію та узгодженість між агентами. Вони особливо ефективні у системах, де критичним є правильне трактування інформації, наприклад в експертних системах. Водночас побудова та підтримка онтологій є трудомісткою, що ускладнює розробку великих проєктів.

У нормативних підходах агенти розглядаються як суб'єкти, що підпорядковуються соціальним правилам, зобов'язанням та санкціям. Методології

OperA та Normative MAS дозволяють моделювати поведінку в умовах складних соціальних взаємодій, що важливо для економічних, правових та переговорних систем. Головна перевага полягає в можливості формалізувати взаємодію між агентами з точки зору норм та відповідальностей. Недоліком є складність побудови й узгодження таких норм у великих динамічних системах.

BDI-підхід моделює агентів як раціональних суб'єктів, які мають вірування (beliefs), бажання (desires) та наміри (intentions). Він забезпечує глибоке та гнучке моделювання індивідуальної поведінки, дозволяючи агентам планувати свої дії. Платформи на зразок Jadex і PRS широко застосовуються у реальних системах, що потребують раціонального планування. Серед мінусів — висока обчислювальна складність і важке масштабування у великих популяціях агентів.

Swarm-методи надихаються природними колективними системами, такими як колонії мурах чи рої птахів. Система визначається набором простих локальних правил, які забезпечують складну глобальну поведінку без централізованого керування. Такий підхід відзначається високою масштабованістю та стійкістю до відмов. Однак глобальна поведінка може бути непередбачуваною, що ускладнює контроль та гарантії безпеки.

Формальні та математичні методи використовують формальні логічні, математичні чи ігрові моделі для опису й аналізу поведінки агентів. Використання строгих формалізмів дозволяє перевіряти властивості системи, гарантувати коректність та знаходити оптимальні стратегії взаємодії. Формальні методи особливо важливі у критичних системах, де безпека та коректність мають першочергове значення. Однак їх застосування вимагає значних обчислювальних ресурсів і високого рівня математичної підготовки.

Методи агентного моделювання у симуляційних середовищах передбачають побудову моделей БПС у спеціальних симуляційних платформах, таких як NetLogo, Repast чи AnyLogic. Вони дозволяють аналізувати емерджентну поведінку великих популяцій агентів і ефективно використовуються в дослідженнях та навчанні. Їх перевага, це швидке створення моделей, що дає змогу експериментувати з різними

сценаріями. Проте такі симуляції не завжди відображають обмеження реальних систем і можуть бути непридатними для прямої реалізації.

Для того, щоб виявити особливості компонентів БПС, ми проаналізували літературу та перерахували наступні ознаки, які використовуються для опису кожного компонента БПС.

Багатокомпонентні програмні системи, як правило, пов'язані з відповідним підходом до наступних тем:

– онлайн-трейдинг та онлайн-торгівля де агенти можуть вивчити та застосувати кілька стратегій торгів для мінімізації витрат та максимізації прибутку. Таким чином, наймання агентів в якості трейдерів може збільшити дохід;

– реагування на стихійні лиха де зазвичай припускають, що агентами керує людина. Вони показують важливу інформацію про місце, де сталося лихо. Однак можна використовувати організаційні можливості агентів у роботах для виконання завдань з ризиками;

– моделювання соціальних структур, яка приносить користь індивідуальній спеціалізації людини і, отже, структурам суспільства, абстрагуючи такі існуючі структури, як інститути, людські робочі групи, команди тощо, для створення систем, натхненних цими структурами. А також моделювати такі соціальні структури та оцінювати ситуації в них.

Перерахуємо деякі з основних застосовних областей агентів:

– система дистанційного керування – найбільш значущий приклад дистанційного керування описаний в роботі, де система, заснована на агентах, виконує дослідження космосу на мільйони кілометрів далеко в космосі. Він виконується за допомогою дистанційного керування з землі, однак, враховуючи дуже велику відстань, існує система на базі MAS, яка повинна приймати важливі рішення за відсутності зв'язку.

– системи підміни людини, які застосовні для небезпечних завдань, такі як дослідження космосу або глибоководдя, протибомбардування або ядерні станції;

– системи імітування поведінки людини. Симуляції емоцій людини виконуються за допомогою агентів;

– системи симуляції, наприклад в кіноіндустрії, де в багатьох стрічках використовують технологію на основі БПС для відтворення сцен масштабних сцен, правдоподібні стрічки, які використовується в комерційних фільмах;

– системи розумних будинків та розумних помічників, в яких використовуєься тенденція до створення БПС, які працюють як помічники, наприклад, для створення веб-сервісів, де людина-користувач може запросити набір агентів для створення композитного сервісу на основі такого запиту. Агент, який вчиться на взаємодії користувача з додатком, а потім імітує його, щоб зменшити кількість роботи та інформаційного перевантаження. Фактичний стан і майбутнє таких інтелектуальних помічників розглядається на сьогодні;

– системи електронної комерції, де такі стратегії використовуються агентами для автоматичного виконання різних ставок та оптимізації співвідношення витрат і вигоди;

– автоматизовані системи в виробництві, де рішення на основі БПС у виробництві з кожним разом збільшуються, надаючи нові способи вирішення проблем ефективності виробництва;

– системи управління бізнес-процесами, де запропонований підхід, заснований на агентах, який показує, як агентська технологія може підвищити ефективність, забезпечуючи краще планування, виконання, моніторинг і координацію ділової діяльності тощо.

Таким чином, узагальнюючи працюючі рішення та різні дослідження термін багатокомпонентних програмних систем постає як інноваційна та ефективна парадигма моделювання штучних складних систем. Багатоагентний підхід успадкований від різних біологічних, соціальних теорій і теорій прийняття рішень. Багатоагентні сутності можуть формуватися в різних видах здібностей від найбільш реактивних до найбільш когнітивних. Такі сутності інтегруються в різні моделі, що операціоналізують архітектуру агентів, можливості взаємодії та сприйняття, а також обробку в середовищі. З соціальної точки зору мультиагентні пристрої складаються з різних моделей взаємодії і організаційних структур, пов'язаних з природою агентів.

Проектування системи на основі мультиагентної концепції є недавнім підходом у галузі програмної інженерії, однак для його реалізації було розроблено безліч методів. Вони були розроблені з метою стати основою для розробки програмного забезпечення, що дозволяє найпростішим способом проектування мультиагентної системи. Існуючі методи, як правило, базуються на проблемі, яка повинна бути вирішена в області інженерії, іншими словами, на задану проблему, яка повинна бути вирішена в рамках певної області, на яку орієнтовані методи. Справа в тому, що ми не знаємо, чи правильний метод домену, який призначений для вирішення проблеми. Це питання, до якого потрібно поставитися уважно.

1.3 Постановка задач та вимог

Сучасним програмним системам притаманна висока складність, зумовлена їх розподіленістю, багатокомпонентністю та необхідністю функціонувати в динамічних середовищах (мобільні застосунки, корпоративні інформаційні системи, системи домашньої автоматизації тощо). У таких умовах традиційні методи проєктування нерідко виявляються недостатніми, оскільки не забезпечують потрібної гнучкості, масштабованості, автономності та здатності системи до адаптації.

Особливу складність створює початковий етап проєктування багатокомпонентних програмних систем: розробник і проєктувальник повинні обрати методологію та архітектурний підхід із великої кількості можливих варіантів, які суттєво відрізняються як за областю застосування, так і за припущеннями щодо середовища, типів агентів, механізмів координації, рівня формальності, підтримки знань (онтологій) та ін. Така ситуація породжує початкову невизначеність і залежність якості рішень від досвіду та попередніх знань проєктувальника. Невдалий вибір методології або неправильне узгодження вимог задачі з можливостями конкретного підходу (наприклад, Gaia, ADELFE, MaSE,

Tropos тощо) може призвести до значного ускладнення розробки, зростання ризиків помилок і навіть до ситуацій, коли подальша реалізація системи стає проблемною.

У зв’язку з цим актуальною є потреба у підтримуючій альтернативі, яка б поєднувала:

– аналіз проблемної області та формалізацію характеристик задачі (робота агента, як специфікація застосунку);

– аналіз та опис компонентів рішення (мета-моделі, онтології компонентів, моделі агентів).

На основі цього забезпечувала обґрунтований вибір найдоцільнішого підходу проєктування й архітектурної інтеграції для конкретної багатокомпонентної системи.

У межах даного дослідження пропонується розв’язати зазначену проблему шляхом створення методу проєктування ПЗ на основі агентно-орієнтованої архітектури та відповідного механізму узгодження, який використовує два ключові джерела даних:

– опис проблеми / вимоги застосунку, що може містити неповну та невизначену інформацію і потребує процедури характеризації (виявлення домену та суттєвих ознак задачі навіть за часткових даних);

–• онтологію та опис компонентів рішення та мета-моделей, що має нижчий рівень невизначеності й може опрацьовуватись засобами семантичного аналізу (через OWL/UML-подання та зв’язки ознак з доменами).

Отже, центральною задачею є розробка механізму, який на ранньому етапі (перед реалізацією) забезпечить автоматизоване зіставлення характеристик задачі з характеристиками доступних мета-моделей і компонентів, а також запропонує проєктувальнику найбільш релевантні варіанти проєктного рішення в межах агентно-орієнтованої архітектури для БПС.

Перспективним напрямом розвитку запропонованого підходу є використання динамічної композиції веб-сервісів, що дозволить модернізувати систему на рівні агентів: додавати нових агентів (і нові функції/сервіси) без перебудови всієї

системи, розширюючи систему за межі розпізнавання текстових шаблонів. Додатково підхід може бути підсилений стандартизованим поданням описів специфікації застосунку (зокрема, XML-орієнтованими форматами на кшталт ADMACA), що робить процес характеризації більш відтворюваним та корисним для інженерів ПЗ як частина попередньої фази розробки.

Таким чином, дана робота розглядається як перший крок до побудови повноцінної попередньої фази проєктування багатокомпонентних програмних систем на основі агентно-орієнтованої архітектури, де ключовим внеском є механізм узгодження між специфікацією задачі та онтологічно/семантично описаними компонентами рішення, що зменшує початкову невизначеність вибору методології та підвищує керованість процесу проєктування.

1.4 Висновки

У першому розділі магістерської роботи виконано характеристику предметної області багатокомпонентних складних систем та обґрунтовано актуальність застосування агентно-орієнтованих підходів для їх проєктування. Показано, що БПС являють собою складні програмні або апаратно-програмні утворення, які складаються з множини автономних агентів, здатних сприймати середовище, приймати рішення та взаємодіяти між собою. На відміну від централізованих систем, глобальна поведінка БПС формується, як результат колективної взаємодії агентів, що забезпечує високий рівень гнучкості, природну масштабованість і стійкість до відмов. Наголошено, що саме децентралізована організація та паралельність обробки дозволяють ефективно використовувати БПС у складних розподілених задачах, зокрема в робототехніці, логістиці, управлінні транспортними мережами, фінансових симуляціях та інтелектуальних системах підтримки прийняття рішень.

Окрему увагу приділено здатності БПС до адаптації та самоорганізації. Установлено, що в умовах динамічних і частково невизначених середовищ агенти можуть коригувати власну поведінку з урахуванням змін середовища та дій інших агентів, що спричиняє виникнення колективного інтелекту та emergent-ефектів.

Саме ці властивості роблять БПС інструментом для прогнозування, оптимізації й розподіленого керування, однак водночас ускладнюють формальне описання, верифікацію та забезпечення передбачуваності поведінки системи.

У розділі систематизовано ключові переваги БПС: відсутність єдиного центру керування, масштабованість шляхом додавання агентів без істотної перебудови архітектури, відмовостійкість, адаптивність, природність моделювання багатосуб'єктних процесів, висока продуктивність завдяки паралельній роботі та можливість комбінування різних стратегій і алгоритмів у межах однієї системи. Разом із тим визначено основні недоліки та ризики, що супроводжують розробку БПС: складність проєктування і координації дій агентів, комунікаційне та обчислювальне навантаження, непередбачуваність emergent-поведінки, проблеми тестування й верифікації через велику кількість сценаріїв взаємодії, підвищені вимоги до ресурсів і складність реалізації механізмів безпеки в розподіленому середовищі.

На основі аналізу сучасних досліджень зроблено висновок, що існуюче різноманіття методологій проєктування агентних систем: організаційно-орієнтованих, адаптивних і кооперативних, goal-орієнтованих, процесних, онтологічних, нормативних, BDI, swarm, формальних, симуляційних тощо, ускладнює початковий вибір підходу для конкретної задачі. Встановлено, що на практиці рішення часто обираються без належного попереднього аналізу відповідності проблемної області та вимог застосунку можливостям обраної методології, що підвищує ризик помилок і невдач під час реалізації. У зв'язку з цим сформульовано головну проблему розділу: необхідність загального механізму попереднього аналізу та узгодження, який би уніфікував вибір моделей і компонентів рішення залежно від предметної області та цільового застосування.

Загалом, у першому розділі обґрунтовано доцільність розроблення методу проєктування на основі агентно-орієнтованої архітектури та визначено вимоги до такого підходу. Центральною задачею визначено створення механізму, що забезпечує автоматизоване зіставлення характеристик задачі зі характеристиками мета-моделей і компонентів рішення на ранніх етапах проєктування. Це формує

основу для подальших розділів роботи, у яких буде деталізовано підхід до характеризації вхідних даних, побудови онтологічного опису компонентів та реалізації механізму узгодження як інструмента підтримки прийняття рішень під час проєктування багатокомпонентних агентних систем.

Таким чином, основний внесок цієї роботи полягає у розробленні процесу зіставлення характеристик компонентів, абстрагованих від наявних методологій та предметної області, з характеристиками специфікації застосунку, що дає змогу сформувати обґрунтований вибір набору компонентів, які репрезентують рішення на основі БПС для заданої специфікації застосування.

2 МЕТОДОЛОГІЧНІ ПІДХОДИ ДО РОЗВ’ЯЗУВАННЯ ЗАДАЧІ ЗА ТЕМОЮ ДОСЛІДЖЕННЯ

2.1 Основні концепції підходу для багатокомпонентних програмних систем

Наша основна проблема полягає в тому, що існує багато методологій, багато проблем і безліч можливих рішень для БПС. Тому рішення про розв'язання проблеми за допомогою рішення на основі БПС не є простим. Також, враховуючи, що у нас слабкий досвід розробки БПС, це ще більше ускладнюється, а ймовірність відмови при використанні БПС стає більшою. Ми не класифікуємо проблеми і не генеруємо єдину методологію, ми усвідомлюємо, що існує кілька існуючих методик, які добре підходять для вирішення тих чи інших типів завдань. Тому наша мета полягає в тому, щоб дослідити, які характеристики проблем пов'язані з існуючими методологіями, щоб згенерувати пропозицію інженеру, який прагне розробити рішення на основі БПС.

Таким чином, внесок цієї роботи обмежується створенням процесу для відповідності характеристик компонентів, абстрагованих від існуючих методологій і предметної області, характеристикам специфікації застосування, щоб запропонувати вибір групи компонентів, які представляють рішення на основі БПС для специфікації застосування.

Коли системний дизайнер починає створювати БПС, він повинен враховувати методологію БПС. Як правило, кожна методологія надає процеси, компоненти та інструменти для розробки бажаного БПС. Тим не менш, з огляду на велику кількість методик, вибір найбільш підходящої методики стає незрячим вибором. Особливо для новачка, який не знайомий з наданими компонентами методології БПС і з поганим досвідом БПС. Однак такий вибір має вирішальне значення для розгортання рішення на основі БПС.

Крім того, невід'ємною проблемою методологій БПС є зручність використання різних методологій-компонентів. Ось чому ми не можемо брати компоненти з різних методологій для легкого створення рішень на основі БПС. Однак існують зусилля

щодо підходу фрагментів методу, коли різні частини процесу беруться з різних методологій, тим не менш, необхідно адаптувати такі фрагменти, використовуючи стандарт фрагментів методу, щоб зробити їх придатними для використання.

Причина полягає в тому, що зазвичай методології не сумісні між собою, тому їх компоненти не можуть працювати разом без важкої адаптаційної роботи. Нарешті, стає важко думати про можливість повторного використання вже створених рішень, оскільки вони є конкретними рішеннями для конкретних проблем. Наприклад, агент, який стикається з рішенням на основі БПС, не може бути просто використаний для іншого, але пов'язаного рішення на основі БПС. Це пов'язано з тим, що реалізація агента залежить від конкретних взаємодій, середовища та організації, створеної для конкретного рішення на основі БПС. Таким чином, адаптація вже впровадженого агента до абсолютно нового рішення на основі БПС може бути найважчою роботою, ніж початок впровадження нового агента.

Таким чином, питання, що лягли в основу створення нашого підходу, полягають у наступному:

– при виборі методології, яка точно відповідає бажаному додатку, розробник системи повинен мати хороший досвід роботи з концепціями БПС і хороші знання бажаної області застосування.

– проектувальник повинен враховувати труднощі залежності методологій компонентів, наприклад, рішення, яке потребує компонентів з різних методологій.

– таким чином, якість знань і досвіду системного дизайнера впливає на якості процесу розробки БПС і кінцевого продукту.

Ось чому ми вважаємо, що ці етапи розвитку безпосередньо пов'язані зі сферою прийняття рішень. Щоб висвітлити ці питання, ми пропонуємо статтю, яка зосереджена на попередньому етапі, щоб допомогти розробнику системи підтримувати свої рішення на перших кроках створення підходу, який охоплює:

A. Метамоделювання БПС:

– Декомпозиція та повторне використання: Підхід повинен розкладати існуючі та майбутні компоненти методологій у метамоделі, щоб зробити їх незалежними та придатними для повторного використання.

– База знань про характеристику метамоделі: Підхід повинен використовувати характеристику кожної мета-моделі, виявляючи та абстрагуючи її ознаки.

– Набуття знань або досвіду: Підхід повинен використовувати статистичні значення для розміщення кожної функції мета-моделі в одній або декількох областях рішення. Таким чином, розглянемо такі ознаки як надійні або ні в межах кожної області відповідно до досвіду (зберігаються у вигляді статистичних значень). Ці значення оновлюються з урахуванням відгуків розробника системи.

– Обмеження мета-моделей: Підхід повинен враховувати питання сумісності між кожною мета-моделлю, щоб уникнути використання несумісних наборів мета-моделей. Крім того, він повинен пропонувати обгорткові рішення, коли це можливо.

Б. Специфікація застосування як опис проблеми:

– Вхідні дані специфікації програми: Підхід повинен отримувати як вхідні дані користувача текстовий або напівструктурований документ, що містить опис проблеми. Такий опис проблеми повинен бути написаний в контексті програмної інженерії та з визначенням того, який стандарт або протокол (наприклад, UML) був використаний.

– База знань з характеристикою проблеми. У підході необхідно використовувати проблемні характеристики абстракцій.

– Набуття знань або досвіду: Підхід повинен використовувати статистичні значення для знаходження кожної характеристики проблеми від однієї до однієї або декількох областей рішення. Іншими словами, кожна характеристика проблеми має відношення з однією або декількома областями рішення, отже, таке відношення має статистичне значення, яке визначає, чи пов'язана характеристика проблеми в області розв'язку чи ні.

– Процес характеристики специфікації застосування. Підхід повинен отримувати вхідні дані користувача та характеризувати їх, використовуючи їхні бази знань для виявлення характеристик проблеми і, отже, домену або доменів рішення.

B. Відповідність специфікації програми та мета-моделям:

– Процес зіставлення. Підхід повинен узгоджувати характеристики проблеми та предметну область з характеристиками метамоделей та спонукати найнадійніші метамоделі до побудови рішення.

– Характеристики: Мета-аналіз: Цей підхід потребує процесу мета-аналізу для вирішення проблем прийняття рішень у процесі зіставлення.

Найважливішим внеском всього проектного підходу є підтримка прийняття рішень розробниками MAS, однак дана дипломна робота зосереджена на останньому пункті, перерахованому вище. Таким чином, основним внеском цієї дисертації є визначення нового підходу до метааналізу, що виконується когнітивними агентами в рамках MAC, який дозволяє приймати рішення, використовуючи характеристики проблеми та особливості метамоделі як статистичні значення. Крім того, кожен когнітивний агент володіє внутрішнім модулем для виконання мета-аналізу, побудованого на баєсовому алгоритмі пізнання.

Основною метою даного проекту є зниження складності та зниження ризиків з використанням мультиагентної системи як основи для вирішення проблеми. Зробіть перші кроки процесу розробки простішими, легшими та безпечнішими для розробників та дизайнерів систем. Як наслідок, зробити мультиагентний системний підхід більш прийнятним для галузі та розширити його використання.

Для того, щоб прояснити процес прийняття рішень агентами і перш ніж представити наш внесок, ми повинні визначити термін мета-аналізу, як ми його розуміємо: мета-аналіз збирає статистичні результати пов'язаних досліджень. У медичній галузі фахівець у цій галузі зазвичай вибирає такі пов'язані дослідження з баз даних, як Medline і PubMed - де такі дослідження зазвичай зберігаються - для прийняття рішень, використовуючи досвід, зібраний у всіх дослідженнях. Цей процес дозволяє використовувати всі попередні аналізи, проведені в цих дослідженнях, для прийняття рішень для поточної справи. Наприклад, мета-аналіз може запропонувати використання певних ліків або методів лікування для лікування хвороби. Отже, щоб скористатися функцією успішного прийняття рішень мета-

аналізом у цій роботі, ми пропонуємо використовувати процес, заснований на метааналізі.

Для того, щоб визначити складну систему, нам потрібно спочатку знати, що таке система. Термін система походить від латинського «systema», а також від грецького «systemat», від терміна «synistanai», поєднаного з «syn-» + «histanai», що означає «змушувати стояти». За словником Мерріам-Вебстера система визначається як: складна єдність, утворена з багатьох, часто різнорідних частин, підпорядкованих загальному плану або служать спільній меті.

Отже, під системою розуміють сукупність взаємодіючих або взаємозалежних компонентів, що утворюють єдине ціле. В інформатиці систему зазвичай пов'язують з програмною системою, побудованою на структурі компонентів, яка має процес зв'язку між такими компонентами.

Термін комплекс походить від латинського «complexus», що означає сукупність і обіймання.

Як зазначено, система є складною, якщо вона має такі атрибути:

А. Склад. Коли складне явище складається з безлічі частин, предметів, одиниць абоокремих осіб.

Б. Інтерактивність. Коли існує багато відносин або виконується багато взаємодій між такими частинами.

В. Поява. Коли ці частини створюють комбіновані ефекти або синергії, які нелегко передбачити і які часто можуть бути новими, несподіваними, навіть несподіваними.

Складна система також розглядається як система, що складається з частин, які пов'язані між собою. Така система в цілому відображає одне або кілька властивостей, не очевидних з властивостей окремих частин. Як правило, поведінка цілої системи серед можливих властивостей не виявляється на індивідуальному рівні.

Також існує кілька видів складних систем:

- Хаотичному. За даними, такі системи чутливі до початкових умов. Це означає, що кожне збурення на початку буде відрізняти майбутню поведінку системи.
- Адаптивна. Її прийнято розглядати як особливу складну систему, яка складається з взаємопов'язаних елементів і володіє навичками збору досвіду і зміни з метою адаптації до навколишнього середовища. Наприклад, адаптивні складні системи ми можемо перерахувати, мозок людини, екосистему, соціальні системи тощо.
- Нелінійних. Цей вид систем зазвичай володіє поведінкою, яка не є предметом суперпозиції. Іншими словами, його поведінка

Система складна різноманітністю і безліччю взаємодій, які вона використовує. Зазвичай такі системи відрізняються від інших неможливістю ідентифікувати всі елементи і зрозуміти динамічно оновлюються взаємодії (Рисунок 2.1.1).

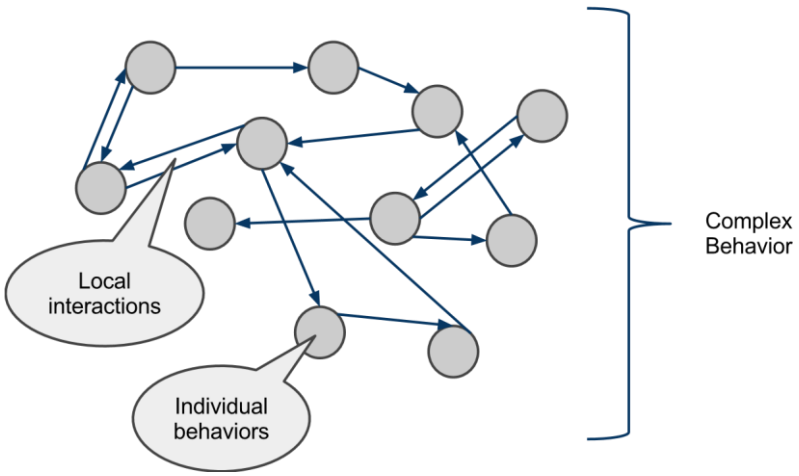


Рисунок 2.1.1 - Складна поведінка, що виникає в складній системі.

Це, як правило, тягне за собою відсутність тотального контролю і незворотність (будь-яка дія не може бути зворотно для зміни динаміки з впевненістю повернення до одного з попередніх рівноважних станів). Складні системи можна розділити на різні рівні взаємодії, що дозволяє додавати до простих елементів більш просунуті компоненти. Ці ж компоненти сприяють виникненню добре організованих та ієрархічних структур, які тісно взаємодіють між собою та з навколишнім середовищем. Структури, що виникають тоді, не можуть бути

зрозумілі просто з використанням сутностей. Складні системи часто є природними і є предметом активного вивчення в таких областях, як фізика, біологія, гуманітарні науки та соціальні та когнітивні науки. У обчислювальних системах системи інформативного, наглядового та розв'язання проблем стають все більш розподіленими, відкритими, великомасштабними та різнокольоровими. Їх взаємозв'язки стають настільки складними і перехрещуються таким чином, що перевершують загальне розуміння, яке може мати справжня людина, роблячи складні штучні системи.

Вимоги щодо в цих системах визначаються за типом агентів. Хоча не існує загальноприйнятого визначення агента. Однак навик автономії - це риса, яка є найбільш прийнятою у визначенні агента. Крім того в різних дослідженнях та сферах використання агент визначається з одніє сторони, як розумна і автономна сутність, яка може виконувати визначене завдання. Таким чином, агент може працювати в середовищі одного агента, вирішуючи індивідуальне завдання, наприклад, розв'язуючи головоломку; або в мультиагентному середовищі, де є два або більше агентів, які виконують індивідуальне завдання та групове завдання за допомогою взаємодії.

Агент також визначається як комп'ютерна система, розташована в деякому середовищі, де він може виконувати автономні дії (а також сприймати зміни) для досягнення своїх цілей проектування.

Слід відмітити, що є різниця між агентом і інтелектуальним агентом:

– агент, це комп'ютерна система, яка розташована в деякому середовищі і здатна до автономних дій в цьому середовищі з метою досягнення своїх цілей проектування;

– інтелектуальний агент вимагає бути реактивним, проактивним і соціальним.

В інших дослідженнях уточняється визначення інтелектуального агента , як такого, що визначає інтелектуального агента як частину програмного забезпечення, яка складається з наступних ознак:

– Розташовані: існуючі в середовищі (Рисунок 2.1.2).

– Автономний: він незалежний, тому не контролюється ззовні.

– Реактивний: реагує на зміни, що сприймаються в навколишньому середовищі.

– Проактивний: послідовно прагне до досягнення нових цілей.

– Гнучкий: має багато способів для досягнення своїх цілей.

– Надійний: відновлюється після невдач і несподіваних ситуацій.

– Соціальні: взаємодія з іншими агентами та навколишнім середовищем.

Рисунок 2.1.2 – Агент автономно взаємодіє зі своїм оточенням.

Тому в даній дипломній роботі ми вважаємо, що система стає складною за кількістю сутностей і потреб у взаємодіях в ній. Також беручи до уваги колективну поведінку, що виникає, пов'язану з такими сутностями.

2.2 Модель програмного ззастосунку на основі агентно-орієнтованої архітектури для багатокомпонентних програмних систем

Враховуючи, що наш підхід вимагає двох джерел інформації, які будуть використовуватися в контексті штучного інтелекту, ми розглянули можливість визначення онтологічного способу зберігання таких даних, включаючи відносини з різними доменами.

Проблематика, коли нам потрібно використовувати онтологічні бази знань, наступна:

1. Характеристика ситуації:
   - характеристика проблеми;
   - домени;
2. Спеціалізовані суб'єкти:
   - мета-моделі;
   - Особливості мета-моделей;
   - домени.

Мета-рівень розташовується на один щабель вище за рівень опису. Відповідно, мета-модель можна визначити як «модель моделі», тобто формалізований опис, який задає правила та конструкції для побудови інших моделей. Інакше кажучи, мета-модель виступає явним набором понять, обмежень і правил, необхідних для створення конкретних моделей у певній предметній області.

Оскільки такі конструкції та регулятивні правила фактично відображають сутності предметної області та зв’язки між ними, мета-модель доцільно інтерпретувати як онтологічну характеристику. Її можна уявити як набір «будівельних блоків», за допомогою яких формуються доменні моделі. У цьому сенсі мета-модель є подібною до онтології, якою користуються моделісти під час проєктування (Рисунок 2.2.1).

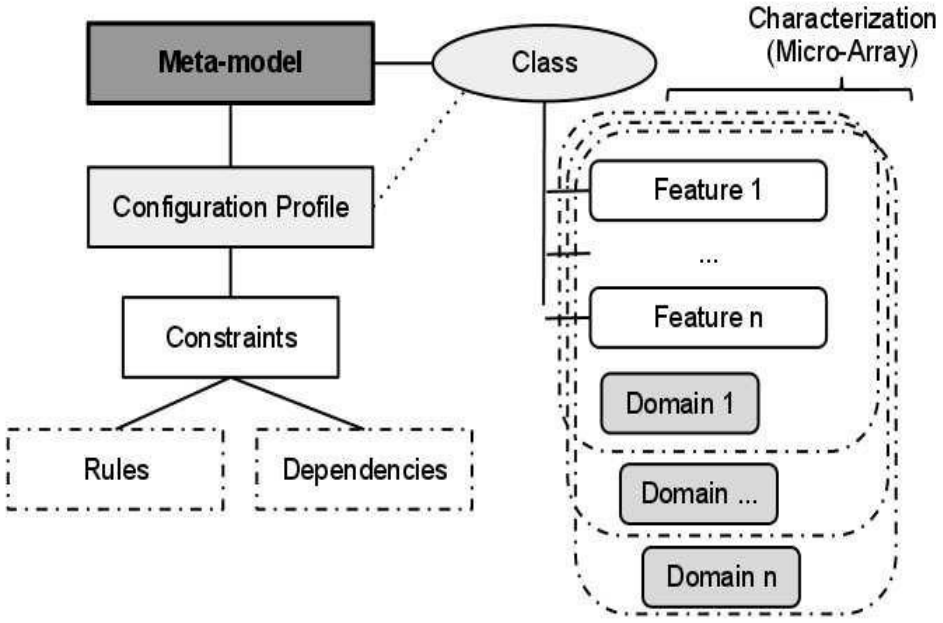


Рисунок 2.2.1 –Мета-модель на основі її характеристик та конфігураційного профілю.

Наприклад, під час використання UML для моделювання програмних систем розробники фактично спираються на онтологічний каркас цієї мови моделювання. UML формалізує базові поняття предметної області, такі як об’єкти, класи, атрибути та зв’язки між ними, а також задає правила їх коректного використання під час побудови моделей. Водночас слід зауважити, що не кожна онтологія обов’язково створюється або подається безпосередньо у вигляді мета-моделі, хоча між цими поняттями існує тісний концептуальний зв’язок.

Для того щоб перетворити опис проблеми на формалізований набір характеристик, інтелектуальна система має виявляти відповідні ознаки на основі наперед визначеного переліку характеристик. Такий перелік повинен зберігатися в онтологічній базі знань, яка виступає джерелом семантично узгоджених понять і властивостей. Саме наявність такої бази знань дає змогу здійснювати коректну інтерпретацію проблемного опису та зменшувати рівень невизначеності на етапі аналізу.

У межах даного дослідження необхідно охарактеризувати опис конкретної проблеми, який визначає цільове застосування системи, тоді як її анотація формується як сукупність ключових характеристик проблеми. Для виявлення цих характеристик доцільно побудувати онтологію, що містить семантично структуровані елементи з урахуванням специфіки проблемних абстракцій. Кожна функція або аспект проблеми має бути віднесена до одного чи кількох відповідних доменів. При цьому домен відносин розглядається як властивість, що використовується для передавання значень ефективності або значущості між елементами моделі.

Під доменом у даному контексті розуміється певна сфера знань, діяльності або впливу, у межах якої формуються та інтерпретуються відповідні поняття. Як було зазначено раніше, описаний етап є початковим у загальному процесі аналізу, а його результат подається у вигляді мікромасиву, який містить інформацію, отриману внаслідок процесу AS-характеристики та специфікації предметної області. Для отримання змістовної інформації з таких мікромасивів виконується подальший пошук і порівняння за аналогічним принципом.

Відмінною особливістю запропонованого в роботі підходу є те, що аналіз тексту здійснюється на різних рівнях — слів, речень і абзаців. Отримані результати відображаються у структурі онтологічного представлення доменного тексту, що дає змогу визначити, який домен є найбільш релевантним і перспективним для подальшого формування проєктного рішення (Рисунок 2.2.2).

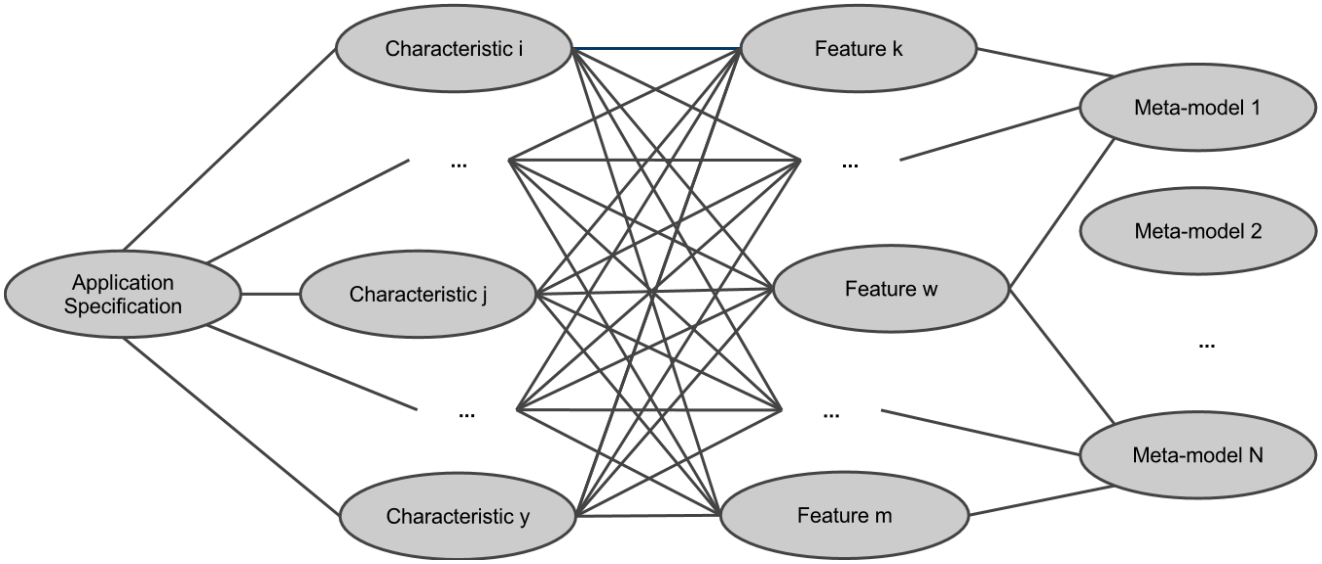


Рисунок 2.2.2 – Відбір кандидатів у мета-моделі за допомогою AS-характеристики

У межах даного дослідження пропонується використовувати мета-моделі, сформовані на основі наявних методологій проєктування із застосуванням підходу голосних AEIO. Такі мета-моделі формуються шляхом абстрагування ключових елементів методологій і можуть бути визначені як у ручному, так і в автоматизованому режимах. Процес абстракції дозволяє виділити суттєві характеристики та усунути надлишкові деталі, зосереджуючись на найбільш значущих аспектах моделі.

У розробленому прототипі особливу увагу приділено спрощенню побудови мета-моделей шляхом використання їх властивостей, заданих у вигляді текстових описів і речень. Такий підхід дає змогу полегшити формування конструкцій мета-моделі та зробити їх більш зрозумілими для подальшого аналізу й обробки.

Відповідно, у роботі було сформалізовано поняття мета-моделі, яке схематично подано на відповідному рисунку.

Запропоновані мета-моделі будуються на основі набору ознак, що характеризують відповідні методології та їхні ключові властивості. Саме завдяки цьому вони можуть розглядатися як кандидати для використання у процесі вибору архітектурного рішення. Вибір здійснюється шляхом встановлення взаємозв'язку між ознаками мета-моделі (ММ-ознаками) та характеристиками застосунку, отриманими в процесі AS-характеристики, як це показано на рисунку 2.2.3.

Процес зіставлення виконується з урахуванням того, що кожна функція системи асоціюється з певним доменом, а зв'язок між ними визначається через значення ефективності. Це дозволяє кількісно оцінити відповідність мета-моделі вимогам конкретної задачі та забезпечити більш обґрунтований вибір оптимального набору компонентів для побудови рішення на основі багатокомпонентних агентних систем.

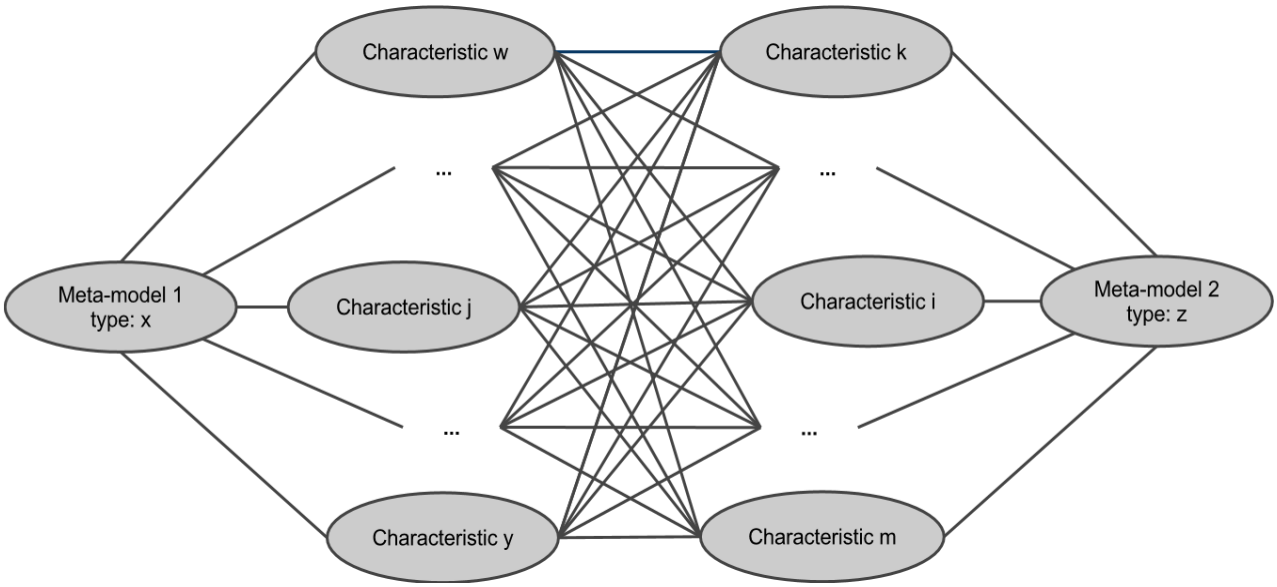


Рисунок 2.2.3 – Об'єднання мета-моделі

Інакше кажучи, у наявних дослідженнях може існувати декілька посилань або інтерпретацій, що стосуються певного домену, однак вибір відповідного домену має бути здійснений заздалегідь – ще на етапі виконання AS-характеристики. Саме тому

база знань у межах запропонованого підходу була сформалізована з використанням стандарту XML, приклад якого наведено в четвертому розділі. Застосування такого стандарту забезпечує структуроване подання знань і спрощує подальшу обробку та розширення. Можливі напрями вдосконалення цього підходу та розвитку механізму зберігання і використання знань детальніше розглядаються в розділі «Подальша робота».

Для формування проєктних рішень із використанням декількох мета-моделей необхідно створити спеціалізовану пам’ять уніфікації, яка схематично подана на відповідному рисунку. Така пам’ять дозволяє встановлювати зв’язки між кожною парою мета-моделей з урахуванням значень ефективності, що відображають взаємну відповідність і узгодженість їхніх характеристик. Завдяки цьому стає можливим інтегрувати різні мета-моделі в єдине рішення, оцінюючи доцільність їх спільного використання.

Перед початком детального опису та реалізації двигуна порівняння необхідно формалізувати низку базових понять, які забезпечують його коректну роботу та надають необхідну інформацію для прийняття рішень. До таких понять належать: специфікація застосування, характеристики проблеми, предметна область, характеристика специфікації застосування, мікромасив, мета-модель, тип мета-моделі, характеристики мета-моделі та характеристика мета-моделі. Визначення цих понять створює концептуальну основу для подальшого функціонування механізму порівняння та узгодження.

Поняття мікромасивів запозичене з галузі біоінформатики, зокрема з аналітичних досліджень дезоксирибонуклеїнової кислоти (ДНК), де вони використовуються для виявлення та аналізу генетичних закономірностей. Мікромасиви являють собою стиснене подання генетичної інформації, яке дозволяє зберігати повний профіль у компактному обсязі даних. У контексті даного дослідження мікромасиви застосовуються для подання характеристик специфікації застосування в межах відповідного домену рішення, а також для формалізованого опису характеристик мета-моделей багатокомпонентних програмних систем, що забезпечує їх подальше порівняння та аналіз.

Характеристика специфікації програмного застосунку полягає в тому, що, як правило, розробка системи розпочинається з етапу збору та аналізу вимог, у результаті якого формується специфікація застосування (AS). Для оцінювання можливості реалізації поставленої задачі за допомогою рішення на основі багатокомпонентних складних систем необхідно визначити ключові характеристики проблеми, закладені в AS, а також відповідну предметну (доменну) область. Виконання такого аналізу вимагає ґрунтовних знань у сфері БПС, а також розуміння особливостей конкретного домену та самої проблеми.

На практиці інженери програмного забезпечення часто стикаються з труднощами, пов’язаними з недостатнім залученням до предметної області майбутнього застосунку або обмеженим досвідом у використанні агентно-орієнтованих підходів. У результаті вибір найбільш придатної методології БПС та її коректне застосування для побудови рішення значною мірою залежать від рівня обізнаності розробника з наявними методологіями та від його практичного досвіду роботи з парадигмою БПС. Саме ця залежність від суб’єктивних чинників є одним із ключових недоліків процесу прийняття рішень на початкових етапах проєктування.

З огляду на зазначене, у даній роботі пропонується розглядати специфікацію програмного застосунку як вхідні дані, надані користувачем. Такі дані доцільно обробляти за допомогою спеціалізованого процесу, який дозволяє автоматично виявляти характерні ознаки специфікації та визначати найбільш відповідну предметну область для сукупності отриманих характеристик. Водночас слід зауважити, що розробка самого процесу розпізнавання домену не є основною метою цієї роботи, а розглядається як допоміжний елемент загального підходу.

На початковому етапі пропонується визначити множину відомих абстрактних доменів та відповідні набори AS-характеристик, які зберігаються в базі знань предметних областей і характеристик застосунків, доступній у базі знань (КВ). У межах дослідження абстракція предметної області подається у формалізованому вигляді, що забезпечує подальший аналіз і зіставлення характеристик специфікації з

відповідними доменами δ_k. Крім того, ми маємо кількість d абстракцій доменів, що зберігаються на КВ наборі D_{KB}:

$$D_{KB} \equiv [\delta_1,...,\delta_d] : \forall k \in [1,d] \quad (2.1)$$

Крім того, ми символізуємо абстракцію з характеристикою AS як κ_k. Також, ми маємо ряд C відомих абстракцій AS-характеристик, що зберігаються в КВ наборі C_{KB}:

$$C_{KB} \equiv [\kappa 1,...,\kappa c] : \forall k \in [1,c] \quad (2.2)$$

Тому абстракція AS-характеристик КВ будується з використанням наступної структури матриць:

$$AS_{KB} \equiv \begin{bmatrix} \theta_{1,1} & \dots & \theta_{1,c} \\ \vdots & \ddots & \vdots \\ \theta_{d,1} & \dots & \theta_{d,c} \end{bmatrix} \quad (2.3)$$

Така матрична структура складається з дійсними значеннями з інтервалу [0,1]. Ці значення пов'язують домени δ_i з AS-характеристиками κ_j. Тому такі значення представляються як $\theta_{i,j}$ і означають відсоток ефективності AS-характеристики κ_j в області δ_i.

Кожен AS проходить процес розпізнавання AS-характеристик, який збігається з AS-характеристиками КВ. Такий процес визначає відомі AS-характеристики, які найкращим чином визначають аналізовану AS. Результат цього процесу зберігається в мікромасиві, який зберігає кодифіковане представлення AS-характеристики. З огляду на процес характеризації, індивідуальний мікромасив AS-характеристики використовує тільки підмножину існуючих характеристик і доменів, що зберігаються в КБ. Таким чином, ми визначаємо C_0 і D_0, як підмножину C_{KB} і D_{KB} відповідно: $C_0 \subseteq C_{KB}$ *i* $D_0 \subseteq D_{KB}$. Тому ми визначаємо кардинальність кожної множини $|C_0| = c_0$ *i* $|D_0| = d_0$, іншими словами, кількість вибраних характеристик і доменів дорівнюють c_0 і d_0 відповідно. Враховуючи, що ми охарактеризували будь-який *ASk* за допомогою процесу AS-характеризації - визначимо єдиний мікромасив AS-характеристики наступним чином:

$$AS'_k \equiv \begin{bmatrix} \theta_{1,1} & \cdots & \theta_{1,c'} \\ \vdots & \ddots & \vdots \\ \theta_{d',1} & \cdots & \theta_{d',c'} \end{bmatrix} \quad (2.4)$$

Де число стовпців - представлене як c_0 - це загальне число AS-характеристик, що співпадають в процесі характеризації AS. Більш того, число рядків - представлено як d_0 - це загальне число доменів, пов'язаних зі знайденими AS-характеристиками. Таким чином, $_{AS0} \subseteq {}_{ASKB,}$ де AS_0 позначає результуючий мікромасив AS-характеристики.

2.3 Модель характеристики мета-моделей

З урахуванням специфіки виконуваного мета-аналізу в роботі пропонується здійснювати формалізовану характеристику мета-моделей та проводити оцінювання їхніх властивостей шляхом аналізу взаємозв’язку кожної окремої ознаки мета-моделі з відповідною областю розв’язання задач. Такий підхід передбачає, що кожна мета-модель описується певним набором ознак, при цьому кожна з цих ознак асоціюється з однією або кількома предметними областями рішення за допомогою показників ефективності, які виступають формалізованим відношенням між ознакою та доменом.

Значення ефективності відображають ступінь придатності та результативності відповідних функцій мета-моделі в межах конкретного домену. Завдяки цьому стає можливим не лише оцінити ефективність окремої мета-моделі щодо певної предметної області, але й виконати порівняльний аналіз кількох мета-моделей або їхніх комбінацій. Така характеристика створює основу для обґрунтованого вибору найбільш доцільного набору мета-моделей для побудови рішення в заданій проблемній області.

Водночас слід зазначити, що отримання коректних і надійних значень ефективності потребує участі кваліфікованих фахівців, які можуть задавати ці показники вручну на основі експертних знань, або ж розроблення автоматизованих механізмів засвоєння знань. Проте питання побудови таких механізмів і організації процесу наповнення знань виходять за межі предмета даного дослідження та не є основним фокусом цієї роботи.

Для визначення абстракцій мета-моделей КВ спочатку зазначимо, що існує чотири різні типи метамоделей (ММ) у наступних наборах КВ:

$$A_{KB} \equiv \{\alpha_1,...,\alpha_a\} \forall k \in [1,a] \qquad E_{KB} \equiv \{\varepsilon_1,...,\varepsilon_e\} \ \forall k \in [1,e] \tag{2.5}$$

$$I_{KB} \equiv \{\imath_1,...,\imath_i\} \ \forall k \in [1,i] \ \ O_{KB} \equiv \{o_1,...,o_o\} \ \forall k \in [1,o] \tag{2.6}$$

Де кожен набір КВ зберігає різні типи ММ: Агент, Середовище, Взаємодія та Організація відповідно. Будь-які ММ α_k, ε_k, $\imath_k$ *і* o_k є різними ММ-типами між собою. Крім того, в кожному типі існують різні види ММ, наприклад, ми можемо мати два різних ММ типу агента як α_1, що представляє «когнітивний агент», і α_2, що представляє «реактивний агент», і ε_0 як «спостережуване середовище» і ε_1 як «3D віртуальне середовище».

Узагальнюючи чотири набори КБ абстракцій ММ, ми визначаємо супермножину КВ Т*КВ* як супермножину КБ, що містить всі відомі ММ-типи наступним чином:

$$T_{KB} \equiv \{\tau_1,...,\tau t\} \forall k \in [1,t] \ TKB \equiv \{A_{KB} \wedge E_{KB} \wedge I_{KB} \wedge O_{KB}\} \tag{2.7}$$

Де кожен τ_k є ММ-абстракцією будь-якого ММ-типу. Це узагальнення супермножини КВ корисне для наступного визначення ММ-характеристики.Ми визначаємо множину F_{KB} як КВ-множину абстракції відомих ознак ММ у наступному вигляді:

$$F_{KB} \equiv [\varphi 1,...,\varphi f\,] : \forall k \in [1, f] \tag{2.8}$$

Таким чином, ми розглядаємо кожне φ_k як абстракцію ММ-ознаки будь-якого ММ-типу.

Для створення КВ ММ-характеристики, кожен ММ повинен пройти процес розпізнавання ММ-ознак – подібний до процесу AS-характеристики, описаного в раніш, але застосованого до ММ щодо домену – який визначає, які відомі ММ-особливості найкраще описують кожну ММ. Однак процес ММ-характеристики виходить за рамки поставлених цілей роботи, проте мінімальний підхід розглядається. Враховуючи, що ми вже охарактеризували будь-який ММτк, результат ММ-характеристики ми констатуємо у вигляді наступної матричної структури:

$$MM'_{\tau_k} \equiv \begin{bmatrix} \mu_{1,1} & \cdots & \mu_{1,f'} \\ \vdots & \ddots & \vdots \\ \mu_{d',1} & \cdots & \mu_{d',f'} \end{bmatrix} \qquad (2.9)$$

Така матрична структура будується з дійсними значеннями, представленими у вигляді $\mu_{i,j}$. Такі дійсні значення взяті з інтервалу [0,1]. Кількість стовпців f_0 – це кількість зіставлених ознак, а число рядків d_0 - це число доменів, пов'язаних зі знайденими ММ-ознаками. Крім того, кожен $\mu_{i,j}$ є значенням ефективності, яке пов'язує кожну знайдену ММ-ознаку φj з кожним пов'язаним доменом δ_i.

Отже, набір КВ, що складається з усіх ММ-характеристик та ознак, має наступну структуру:

$$MM_{KB} \equiv \{MM_{\tau 1} \quad ... \quad MM_{\tau n}\} \qquad (2.10)$$

Попередні структури КВ ММ показують, що вони організовані за ММ-типами, проте домени є основним дискримінуючим фактором. Кожен ММ-тип містить матрицю, де рядки доменів і стовпців – це функції. Таким чином, вибір невеликої кількості доменів дозволяє вибрати лише рядки, пов'язані з такими доменами, тому використовувати лише значення ефективності $\mu_{i,j}$, пов'язані з кожною ознакою в межах цих доменів. Нарешті, в кожному ММ використовується одне і те ж визначення доменів.

Правила об'єднання побудовані у вигляді гіперматриці ознак φ_k відношень. Де кожна ознака пов'язана між собою для кожного окремого ММ-типу. Для індивідуума два ММ-типи виглядають наступним чином:

$$u_{\tau_i,\tau_j} = \begin{bmatrix} & & \tau_i & & \\ & & \varphi_k & \cdots & \varphi_l \\ & \varphi_n & \upsilon_{1,1} & \cdots & \upsilon_{1,m} \\ \tau_j & \vdots & \vdots & \ddots & \vdots \\ & \varphi_p & \upsilon_{m,1} & \cdots & \upsilon_{m,m} \end{bmatrix} \quad (2.11)$$

Крім того, ця структура працює як пам'ять, яка пов'язує функції кожного типу ММ між собою, додаючи значення сумісності. Такі значення отримуються за допомогою автоматизованого рішення для його вивчення або при необхідності можна записати їх вручну - виходячи з досвіду розробника -. Однак ця робота не акцентує на цьому увагу, а використовує це для основної роботи. Ми виконали ці значення та ММ-КВ за допомогою опитування.

2.4 Висновки

У другому розділі обґрунтовано ключову проблему проєктування багатокомпонентних складних систем, яка полягає у значному різноманітті методологій, класів задач і можливих варіантів реалізації. Показано, що вибір рішення на основі БПС не може здійснюватися «за замовчуванням», оскільки кожна методологія є ефективною лише в межах певних доменів і за наявності конкретних характеристик проблеми. Додатковим фактором, що ускладнює цей вибір, є недостатній практичний досвід розробників у галузі БПС, що підвищує ризики помилок та ймовірність невдачі під час впровадження.

Встановлено, що проблема вибору методології для БПС належить до сфери прийняття рішень на ранніх етапах проєктування. Розробник має одночасно враховувати характеристики задачі, предметну область, вимоги до взаємодії агентів та обмеження обраної методології. У разі нестачі доменної експертизи та знань про методології БПС процес перетворюється на «сліпий вибір», який є критично важливим для подальшої реалізації системи. Окремо підкреслено проблему несумісності методологій між собою: компоненти різних підходів важко поєднувати без трудомісткої адаптації, а повторне використання вже створених агентів і рішень обмежене їх залежністю від конкретного середовища, організації та протоколів взаємодії.

У розділі сформульовано підхід, спрямований на зниження невизначеності вибору та зменшення залежності від суб'єктивного досвіду проєктувальника. Основний внесок обмежено розробленням процесу узгодження (зіставлення) між характеристиками специфікації застосунку та характеристиками компонентів/мета-моделей, абстрагованих з існуючих методологій і доменної області. Показано, що підтримка прийняття рішень потребує комплексного попереднього етапу, який включає: метамоделювання методологій БПС із виділенням незалежних і потенційно повторно використовуваних компонентів, подання специфікації застосування як вхідного проблемного опису та її характеризацію через базу знань та процес зіставлення характеристик проблеми та доменів із характеристиками мета-моделей для отримання рекомендацій щодо найбільш придатних компонентів рішення.

Також у розділі запропоновано концептуальну основу зберігання знань у вигляді онтологічно структурованих баз, а результати характеризації подано через формалізовані матричні структури та мікромасиви. Визначено, що зв'язок між характеристиками проблеми, доменами та ознаками мета-моделей може задаватися статистичними показниками ефективності в інтервалі [0;1], які відображають ступінь релевантності функцій у конкретних предметних областях. Наголошено, що отримання таких значень може здійснюватися експертно або через механізми засвоєння знань, однак ці питання не є основним фокусом дослідження.

Загалом зроблено висновок, що запропонована в розділі постановка задачі та формалізація понять створюють підґрунтя для реалізації механізму підтримки рішень у БПС від характеризації специфікації застосування та вибору домену до оцінювання й узгодження набору мета-моделей. Це забезпечує більш керований і обґрунтований вибір архітектурних компонентів, знижує початкову складність проєктування та мінімізує ризики, пов’язані з недостатнім досвідом розробника і несумісністю методологій.

3 ПРОЕКТУВАННЯ АРХІТЕКТУРИ БАГАТОАШЕНТНОЇ ПРОГРАМНОЇ СИСТЕМИ.

3.1 Модуль реалізація програмного застосунку

Процес AS-характеризації не входить до основного предметного поля цієї дисертації. Водночас для забезпечення роботи механізму узгодження (який є центральною фазою дослідження) необхідно окреслити принаймні мінімальний підхід до характеристики текстової специфікації застосування. Тому в цій главі подано базові визначення, потрібні як для реалізації спрощеної AS-характеризації, так і для подальшого функціонування механізму відповідності, а також описано реалізаційний підтип, що виконує таку характеристику. Детальніший опис підходу наведено в наступному розділі, однак підкреслюється, що навіть поверхнева характеристика є достатньою для формування вхідних даних, необхідних для запуску двигуна відповідності.

У межах програмної інженерії пошук ознак, які найточніше описують проблему, зазвичай приводить до розуміння її предметної області. Виявлення характеристик і доменів дає змогу сформувати початкові орієнтири для планування побудови рішення. Проте визначення таких ознак і доменів потребує від інженера практичного досвіду, залученості до домену та знань про нього. Для недосвідчених фахівців або для розробників, які не пов’язані з конкретною предметною областю, коректна характеристика проблеми суттєво ускладнюється. З цієї причини запропоновано мінімальний підхід, що передбачає використання БПС-орієнтованого двигуна для аналізу текстового опису застосунку та автоматизованого виділення характеристик проблеми й доменної специфікації, необхідних для формування потенційного рішення на основі БПС.

Запропонований двигун поєднує архітектуру БПС із механізмом розпізнавання текстових шаблонів та спирається на семантично структуровану базу знань, у якій зберігаються ознаки задачі та елементи предметної області, пов’язані з результатами шаблонного аналізу. Основне призначення такого підходу полягає в тому, щоб виступати як динамічний, розширюваний допоміжний інструмент для інженерів,

який полегшує характеристику проблеми на основі текстового опису бажаного застосування.

Наявність великої кількості мультиагентних методологій створює додаткову складність: розробники часто вагаються у виборі підходу, оскільки кожна методологія використовує власний набір моделей, компонентів і процесів. У результаті проєктувальник має не лише знати властивості мультиагентних моделей, а й уміти узгоджувати їх зі специфікацією застосування та предметною областю. У практичній розробці поширеною проблемою є те, що інженери не володіють достатнім розумінням домену, з яким змушені працювати, тому спершу мають ідентифікувати домен і лише потім здобувати потрібний доменний бекграунд. Така «крута крива навчання» є однією з причин обмеженого поширення мультиагентних методів. Саме тому супутниковий етап, описаний у цьому розділі, має на меті запропонувати мінімальний процес виявлення характеристик із текстового опису специфікації програми, результати якого використовуються як вхідні дані для основного двигуна узгодження.

У наукових роботах описано низку підходів до абстрагування інформації з «сирих» даних, зокрема з використанням WordNet як лексико-семантичного сховища англійської мови. Подібні підходи демонструють можливість систематизації текстових структур і встановлення семантичних зв’язків між термінами, що може бути використано для пов’язування фрагментів тексту зі заздалегідь визначеними проблемними характеристиками. У цьому сенсі запропонований підхід можна інтерпретувати як «перекладач», який перетворює текстовий опис застосування у формат, придатний для машинного опрацювання. Відповідно, як і у статистичному машинному перекладі, доцільним є накопичення статистичних даних, що дозволяють поступово підвищувати якість ідентифікації ознак і доменів. Саме тому пропонується додаткова БПС-архітектура, яка забезпечує розподілення фази AS-характеристики та збір статистичної інформації за аналогією до принципів, що використовуються в сучасних перекладацьких системах.

У межах цієї супутникової фази пропонується окремий БПС-орієнтований процес аналізу текстового опису програми, який виконує інтелектуальне опрацювання змісту для виявлення характеристик проблеми та області її розв’язання. Отримані дані застосовуються для пошуку кандидатних мета-моделей, що вважаються релевантними для виявлених характеристик. Результат аналізу подається у вигляді мікромасиву характеристик опису застосування (ADMACA) – машинозчитуваного представлення, що містить статистичні відомості про виявлені ознаки проблеми та доменні специфікації. Додатково визначається XML-орієнтований стандарт для керування, обміну, повторного використання й обробки результатів ADMACA, що робить дані придатними для накопичення та повторного застосування.

Загалом запропонований процес AS-характеризації виконує допоміжну функцію щодо основної фази узгодження: він «живить» механізм відповідності мікромасивом як машинозчитуваним входом, у поєднанні з характеристиками мета-моделей БПС. Запропонована архітектура містить сукупність агентів з різними ролями та поведінками і поділяється на два взаємопов’язані напрями: БПС та веб-сервіси. У межах БПС-компоненти чотири агенти виконують спеціалізовані завдання, результати яких інтегруються у характеристику опису програми. Кожна така характеристика розглядається як «проба», що відповідає конкретному застосуванню, та формується на основі ознак проблеми, доменних специфікацій і кількісних оцінок у вигляді значень належності (Рисунок 3.1.1).

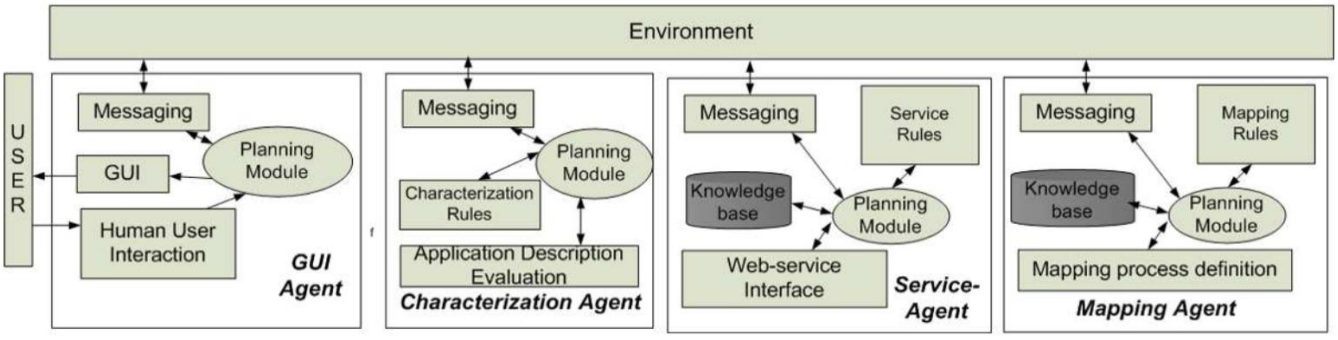


Рисунок 3.1.1 – БПС для архітектури характеризації в цілому.

Ці дані повинні бути відновлені в процесі характеризації з використанням інформації, наданої завданнями агента. Отже, коротко охарактеризуємо завдання кожного агента в наступних рядках:

– Агент з графічним інтерфейсом. Їх основне завдання полягає в тому, щоб зв'язатися з користувачем і він приймає введені користувачем дані і управляти ними за допомогою системних відповідей користувачеві. У нашому випадку ми визначили як вхідні дані користувача текст опису програми. Цей агент керує вхідним вмістом і отримує запитання або результати до користувача, якщо вхідні дані містять недостатньо інформації, він запитує більше даних. Таким чином, це означає, що цей агент може зв'язатися з користувачем, щоб попросити додаткову інформацію, якщо вона потрібна. Таким чином, структура графічного агента включає в себе графічний інтерфейс користувача, напівдіалоговий інтерфейс користувача з управлінням введенням-виведенням користувача, і, нарешті, поведінку на основі повідомлень, яка дозволяє отримувати повідомлення від решти агентів і перетворювати їх в зрозумілу людині інформацію.

– Характеристика агента. Його основне завдання полягає в управлінні процесом характеризації, тому ми повинні визначити процес для характеристики введених користувачем даних. Тому цей агент володіє когнітивною поведінкою, яке керує і планує процес характеризації.

– Основне завдання сервісного агента полягає в управлінні підключенням веб-сервісу. Таким чином, цей агент отримує запит від агента з характеризації з проханням виконати деякі завдання, пов'язані з діяльністю по характеристиці. Він працює як агент, який прозоро керує веб-сервісом, щоб надавати послугу як агент. Таким чином, він дозволяє поліпшити або змінити web-сервіс і тільки модифікувати агента, який надає цю послугу.

– Агент відображення. Цей агент виконує ідентифікацію та відображення даних (наприклад, текстових шаблонів), отриманих від агента характеризації. Це здійснюється за допомогою когнітивного прийняття рішень, які зіставляють дані з існуючими даними в базі знань, які можуть бути визначені так, як ми пропонуємо в онтології особливостей задачі. У нашому випадку ми пропонуємо, щоб відповідна

інформація про проблемні особливості та онтології специфікації домену відновлювалася через Ontology Connection Agent і зберігалася тимчасово в базі знань Problem Features Detector Agent та Domain Locator Agent, обидва вважаються подібними до цього виду агентів.

Використання цих видів агентів у процесі характеризації дозволяє нам створити рішення на основі БПС, для характеристики опису програми та іншого виду подібного завдання характеризації. У цьому додатковому підході ми пропонуємо як приклад використання тексту в описі програми, а для характеристики даних пропонуємо використовувати розпізнавання текстових образів. У зв'язку з цим, зокрема, ми пропонуємо впровадження наступних агентів:

– Особливості детектора агента. Цей агент виконує виявлення проблемних особливостей за допомогою текстових шаблонів, отриманих від агента аналізу тексту. Це здійснюється за допомогою когнітивного прийняття рішень, які зіставляють шаблони тексту з існуючими ознаками, визначеними в онтології ознак задачі. Інформація про онтологію проблемних особливостей відновлюється через Ontology Connection Agent.

– Агент з пошуку доменів. Цей агент знаходить можливу область опису програми, використовуючи шаблони текстової структури, проблемні особливості, пов'язані з шаблонами, та існуючі домени через агент зв'язку онтології. Таким чином, цей агент виконує основне когнітивне завдання з індукції можливого домену або доменів для шаблонів структури тексту, отриманих від агента аналізу тексту.

– Агент Perl. Основне завдання полягає в управлінні підключенням веб-сервісу скриптів на Perl. Таким чином, цей агент отримує запит від агента аналізу тексту з проханням проаналізувати повний або частково текст. Крім того, цей агент керує правилами регулярних виразів (регулярних виразів), які використовуються в процесі. Він працює як агент, який прозоро керує веб-сервісом, щоб надавати послугу як агент. Таким чином, він дозволяє поліпшити або змінити web-сервіс і тільки модифікувати агента, який надає цю послугу.

– Агент зв'язку онтології. Основне завдання полягає в управлінні веб-сервісом онтології підключення. Отже, цей агент отримує запит від агентів

Детектора Функцій та Локатора Доменів для запиту онтологій. Таким чином, він працює аналогічно тому, як агент Perl надає інтерфейс агента веб-сервісу.

3.2 . Специфікація програмного застосунку

З метою продемонструвати мінімально життєздатний робочий приклад у межах дослідження було визначено базовий процес розпізнавання тексту, який розглядається в цьому розділі. При цьому слід підкреслити, що завданням роботи не є розроблення нового або альтернативного підходу до текстового аналізу. Запропонований механізм розглядається виключно як допоміжний інструмент, що функціонує в контексті процесу AS-характеристики та забезпечує отримання вхідних даних для подальших етапів узгодження.

Процес розпізнавання тексту ґрунтується на багаторівневому аналізі, спрямованому на вилучення релевантної інформації з текстового опису. Для реалізації характеристики даних пропонується розділити повний текстовий документ на три основні рівні аналізу: рівень абзаців, рівень речень та рівень окремих слів. Такий поділ дозволяє сформувати композиційну структуру текстових шаблонів, яка за своєю логікою є подібною до баєсової мережі.

Вибір саме такої топології зумовлений її здатністю чітко розмежовувати дані, зберігаючи при цьому контекст та ієрархічний характер текстового аналізу. Запропонований підхід підтримує два напрями обробки інформації — зверху вниз і знизу вгору, що реалізується за допомогою відповідних скриптів на мові Perl, натхненних ідеями статистичного та шаблонного аналізу тексту в процесі видобутку тексту. Крім того, Баєсова природа висновування пов'язана з прийняттям рішень за допомогою оцінки значень ймовірностей зі статистичною та історичною інформацією для протистояння невизначеності. Ця функція є ситуативною для нашого підходу до штучного інтелекту, оскільки ми керуємо описом програми, наданим людиною-користувачем. Отже, першим кроком алгоритму аналізу тексту є спуск по структурі тексту, починаючи з верхнього рівня документа, розбиваючи

його на абзаци; потім кожна гілка абзацу поділяється на речення; Отже, кожне речення ділиться на слова (Рисунок 3.2.1).

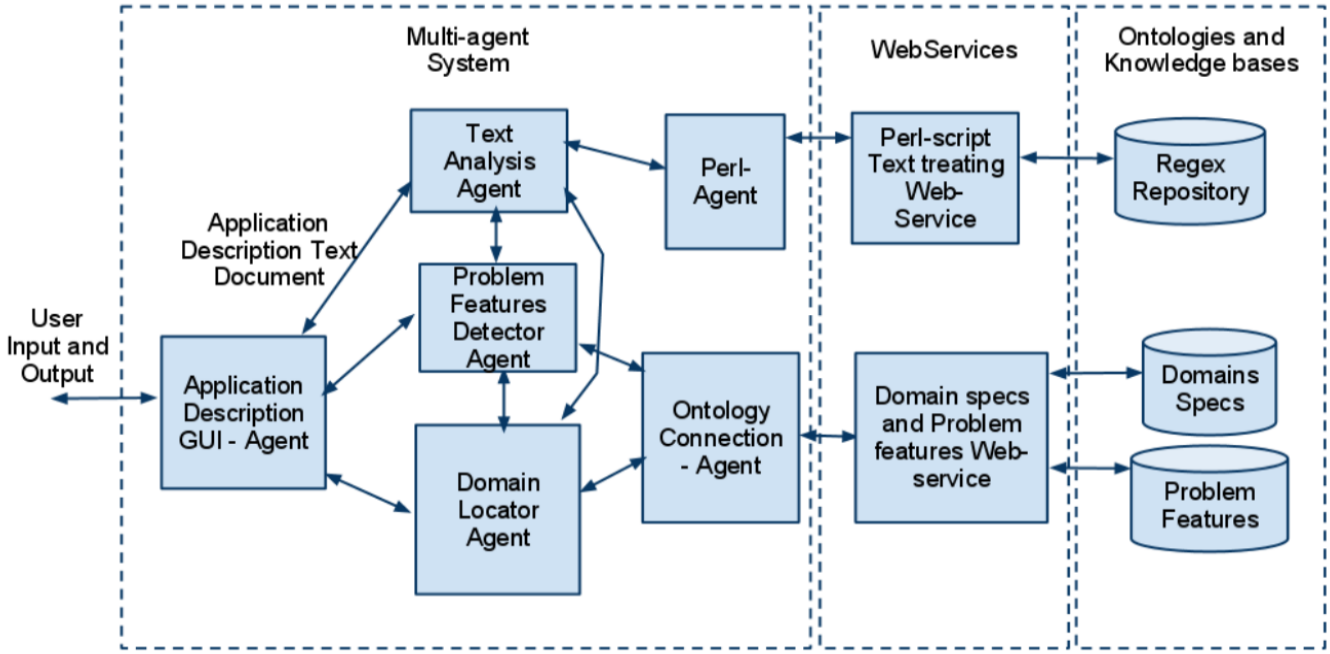


Рисунок 3.2.1 – Приклад характеристики на основі розпізнавання текстових образів

На цьому кроці ми спустилися в структуру тексту аналогічним чином, як показано на Рисунку 3.2.2. Формально ми визначаємо кожен набір слів, речень та абзаців наступним чином:

$$W \equiv \{\omega_1,\ldots,\omega_w\}\,\forall k \in [1,w] \tag{3.1}$$

Де кожне ω_k – допустиме слово, яке зустрічається в описі програми. Ми використовуємо підмножини цих слів для створення речень. Таким чином, множина речень складається таким чином:

$$S \equiv \{\sigma_1,\ldots,\sigma_s\}\,\forall k \in [1,s] : \sigma_k \subset W \tag{3.2}$$

Використовуючи підмножини речень, ми будуємо абзаци, які формально визначаються в абзац, поставлений таким чином:

$$G \equiv \{\rho_1,\ldots,\rho_g\}\,\forall k \in [1,g] : \rho_k \subset S \tag{3.3}$$

Також ми використовуємо набір реальних значень, де кожне з них представляє результат аналізу тексту опису програми у вигляді значення співвідношення для кожної пов'язаної структури тексту:

$$V \equiv \{v_1,\ldots,v_v\}\,\forall k \in [1,v]\; v_k \in R,[0,1] \tag{3.4}$$

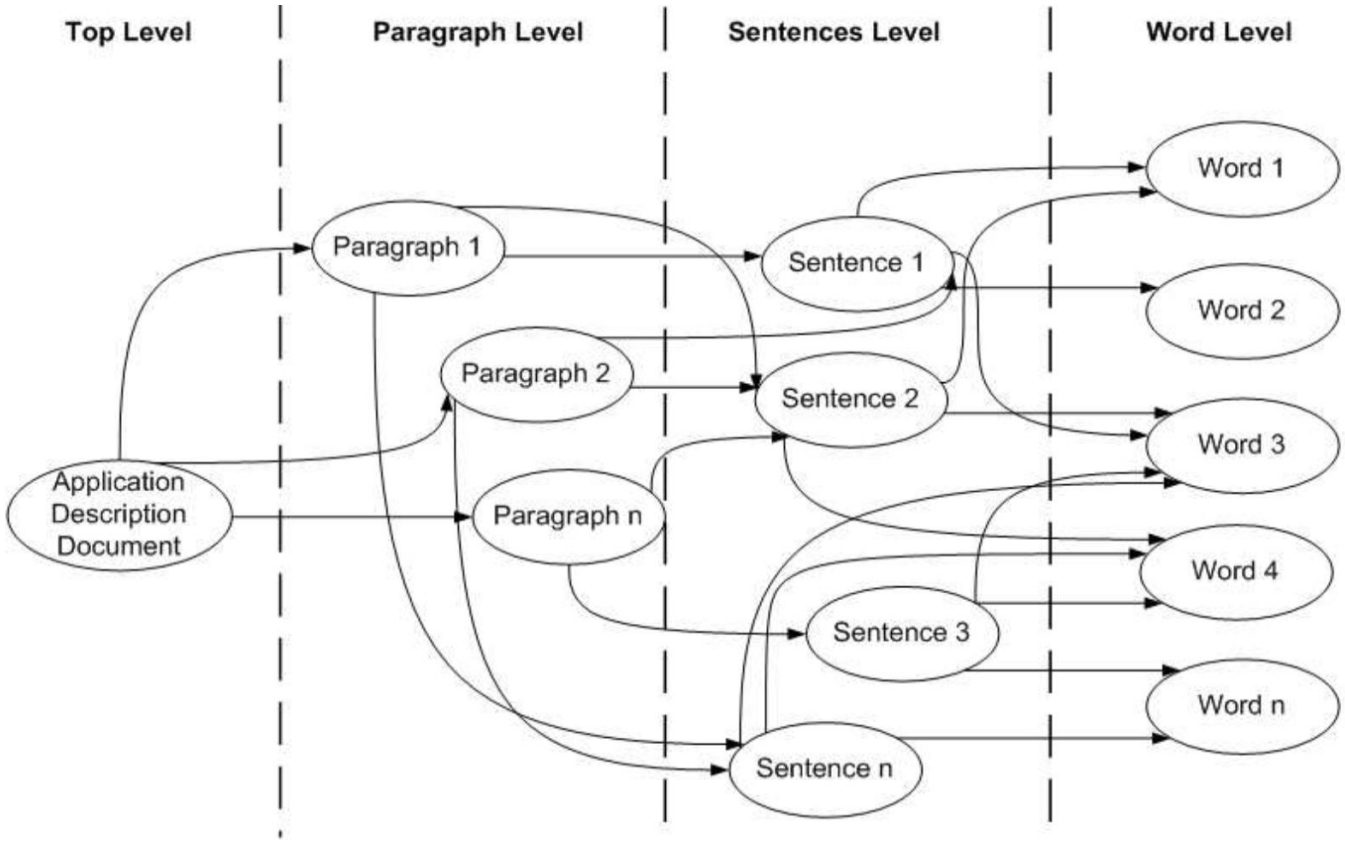


Рисунок 3.2.2 – Процес розпізнавання текстових образів.

Тому другим кроком є фільтрація рівня слів, залишивши лише ключові слова; включаючи власні назви, прикметники та дієслова (усі варіанти: відмінюваний, інфінітив тощо), але відкидаючи займенники та сполучники. Потім ми даємо значення кожному ключовому слову, виконуючи операцію співвідношення з отриманої підмножини ключових слів останнього кроку та підрахунок входження слів кожної підмножини в підмножині, ділення її на загальну кількість слів у підмножині.

Фрагмент описаного результату показаний нижче, де ми бачимо стандарт на основі XML, який визначає приклад знайдених шаблонів ключових слів. Тег кожного ключового слова містить теги з входженнями, де кожне входження має ідентифікатор абзацу та речення, де його було знайдено.

```
<word value="auction" overall =0.0573248407643312 >
    <occurence id="1" para_id="1" sen_id="2" />
    <occurence id="2" para_id="1" sen_id="4" />
    <occurence id="3" para_id="2" sen_id="6" />
    <occurence id="4" para_id="3" sen_id="10" />
    <occurence id="5" para_id="3" sen_id="11" />
    <occurence id="6" para_id="3" sen_id="11" />
    <occurence id="7" para_id="3" sen_id="11" />
    <occurence id="8" para_id="3" sen_id="11" />
    <occurence id="9" para_id="3" sen_id="11" />
    <occurence id="10" para_id="1" sen_id="4" />
    <occurence id="11" para_id="3" sen_id="10" />
    <occurence id="12" para_id="3" sen_id="11" />
</word>
<word value="automated" overall =0.00636942675159236 >
    <occurence id="1" para_id="3" sen_id="10" />
</word>
<word value="available" overall =0.00636942675159236 >
    <occurence id="1" para_id="1" sen_id="2" />
</word>
<word value="behavior" overall =0.0127388535031847 >
    <occurence id="1" para_id="2" sen_id="6" />
    <occurence id="2" para_id="2" sen_id="7" />
</word>
<word value="biders" overall =0.0127388535031847 >
    <occurence id="1" para_id="1" sen_id="4" />
    <occurence id="2" para_id="2" sen_id="6" />
    <occurence id="3" para_id="2" sen_id="8" />
</word>
```

Наступний крок піднімається за шаблонами ключових слів з рівня ключового слова на рівень речення, використовуючи значення значущості ключових слів, ми призначаємо нове значення значущості для кожного речення. Ми беремо кожне зі значень слів, пов'язаних із реченням, щоб підсумувати їх усі та отримати середнє значення слів, пов'язаних із реченням; Тоді ми беремо середнє значення як значення значущості речення. Ми повторюємо цей крок з кожним реченням, пов'язаним з абзацом.

Нарешті, результатом є текстова багатошарова структура шаблону, зважена. Така структура дозволяє досліджувати його, починаючи з будь-якого шару. Це означає, що ми можемо зіставляти схожі структури тексту, щоб побачити, чи є схожі речення або абзаци, використовуючи ключові слова, але оцінюючи загальну на рівні речень, або абзаців (Рисунок 3.3.3). Цей підхід до розпізнавання текстових образів корисний у двох основних аспектах.

По-перше, автоматизоване навчання з використанням інженерії знань дозволяє зберігати структури текстових шаблонів з їх значеннями, що пов'язують їх з особливістю задачі або специфікацією предметної області.

По-друге, для автоматизованої характеристики ми можемо оцінити вхідні описи додатків, використовуючи історичні результати, збережені в першій частині. Це надасть нам автоматизований процес характеристики особливостей проблеми та специфікацій домену.

Тим не менш, у цьому додатковому підході ми зосереджуємося лише на другій стороні, де текстові шаблони знаходять і характеризують за допомогою движка MAS. Наша мета – показати, як ми можемо надати можливе рішення за допомогою MAS для характеристики опису програми. Однак можна використовувати і інший метод, ніж розпізнавання образів в текст.

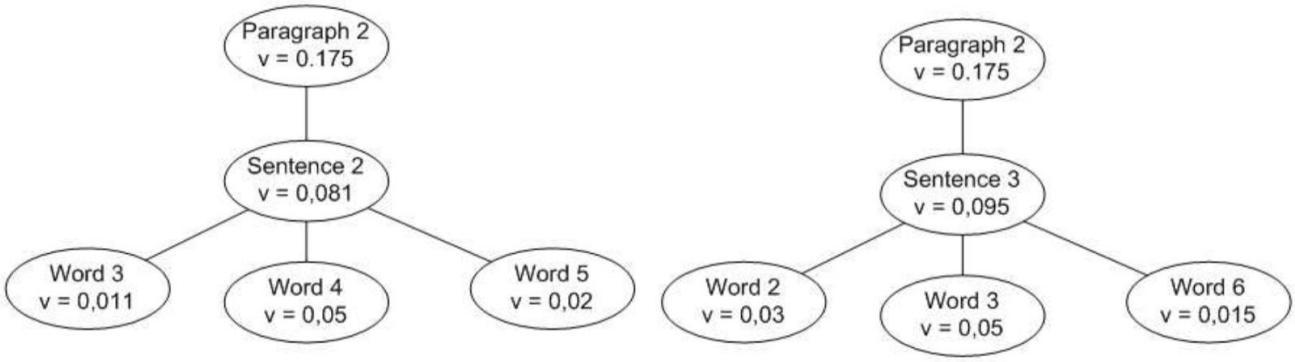


Рисунок 3.3.3 – Графічне зображення шаблонів тексту з відносинами на рівні речення

Отже, це завдання виконує агент Perl і результатом є текстові шаблони. Ці патерни зіставляються з проблемними особливостями та аналогічними онтологіями доменів.

Використання стандарту для реєстрації результатів надає нам можливість зберігати, керувати та повторно використовувати аналізи, виготовлені за допомогою описаного підходу. Однією з головних причин створення цього стандарту є надання результатів у вигляді сховища незалежних, але пов'язаних аналізів у контексті характеристики з метою залишити дорогу готовою до приходу додаткового процесу метааналізу або видобутку даних. Отже, ми визначили стандарт на основі XML, який включає в себе наступні розділи даних:

– Дані профілю користувача. Він містить особисту та контактну інформацію користувача, а також інформацію про установу, якщо це необхідно. Приклад коду фрагмента файлу дивіться нижче:

– Дані профілю користувача фрагмент з прикладу файлу, записаного за стандартом ADMACA

```
<профіль>
<розробник>
<повне ім'я>
<ім'я>Іван</ім'я>
<прізвище>Петров </прізвище> Васильович
</прізвище, ім'я, по батькові>
<організація>
<id>1</id>
<назва>Університет народний - </назва>
<опис>Розробка групи складних систем</опис>
</організація>
<електронна пошта>
<ім'я користувача>Іван </ім'я користувача>
<сервер>servmas.dom.ua</сервер>
</електронною поштою>
<вебсторінка href="http://ivan.servmas.dom.ua" />
</розробник>
```

```
</профіль>
```

– Протокол збору використовуваних даних. Цей розділ має назву та опис протоколу, що використовується при характеристиці; У нашому випадку ми використовуємо аналіз текстових зразків. Тим не менш, протокол збору може відрізнятися від аналізу текстових зразків. Цей розділ супроводжується складом протоколу маркування; Це обчислюються елементи і отримані параметри кількісного визначення. У нашому випадку протокол зберігає ознаки проблеми та специфікацію предметної області у вигляді відображених елементів з їхнім приналежністю ймовірнісних значень як параметрами кількісного визначення. Дивіться наступний фрагмент файлу як приклад:

Дані протоколу та дані маркування

```
<протокол>
<id>1</id>
<ім'я>Аналіз шаблону тексту</ім'я>
</протокол>
<labeling id="1" name="Характеристика проблемного домену">
<Елементи дизайну>
<element type="Проблемна функція">
<design-id>1</design-id>
<title>Функція максимального посилення</назва>
<опис>Оптимізуйте виграш, знайшовши найнижчу вартість або ціну
</опис>
</елемент>
<element type="Проблемна функція">
<design-id>2</design-id>
<title>Функція ролей на аукціоні</title>
<опис>Актори - це покупці, які виступають у ролі покупців і продавців.
```

```
</опис>
</елемент>
<element type="Специфікація домену">
<design-id>3</design-id>
<title>Автоматичний аукціон</назва>
```

<опис>Автоматизований аукціон складається з акторів, де кожен актор може взяти на себе роль аукціоніста або клієнта; Також покупець може виступати в ролі покупця або продавця.

Фрагмент з прикладу файлу, записаного за стандартом ADMACA:

```
</опис>
</елемент>
</Елементи дизайну>
<Кількісні види>
<>
<quantitation-id>1</quantitation-id>
<title>Актуальність</назва>
<опис>Значення ймовірності приналежності</опис>
</тип>
</кількісні види>
</маркування>
```

Отримані дані характеристики. Результати аналізу зберігаються за допомогою посилань id на визначені елементи та параметри збору в протоколі маркування. Дивіться наступний приклад коду фрагмента:

Дані характеристик фрагмент з прикладу файлу, записаного за стандартом ADMACA:

```
<характеристика id="1">
<value design-id="1" quantization-id="1">0.7845</value>
<value design-id="2" quantization-id="1">0.9312</значення>
```

```
    <value design-id="3" quantization-id="1">0,8923</значення>
    </характеристика>
```

Запропонований підхід було апробовано на основі текстових описів програмних застосунків із багаторазовим коригуванням регулярних виразів з метою підвищення якості розпізнавання текстових шаблонів. Зазначені модифікації виконувалися в межах робочого процесу агента Perl і не потребували перероблення або структурних змін усієї системи. Окрім цього, здійснювалося ручне додавання та вилучення записів у відповідних онтологіях проблемних функцій і специфікацій, що дало змогу перевірити гнучкість механізму управління знаннями.

Отримані результати підтверджують, що реалізований у межах даного процесу підхід на основі багатокомпонентних складних систем є адаптивним та стійким до динамічних змін. Процес аналізу характеристик функціонував коректно та відповідав очікуваній поведінці системи. Водночас варіативність результатів, зумовлена змінами регулярних виразів і вмісту онтологій, є природною для такого класу систем і не розглядається як недолік запропонованого підходу.

Це пояснюється тим, що ключовим завданням процесу характеризації є робота з невизначеністю шляхом виявлення найбільш значущих ознак і підказок із використанням наявних знань і методів аналізу. Крім того, результати первинної характеризації передбачають подальшу обробку в межах додаткових етапів, зокрема процесу метааналізу, де здійснюється повторний аналіз отриманих даних. Завдяки цьому кінцевий вплив локальних варіацій у вхідних параметрах зводиться до мінімуму та не впливає критично на загальну ефективність системи.

3.3 Діаграма модулів ПЗ

Враховуючи архітектуру нашого підходу, ми реалізували прототип програмного забезпечення з використанням Java, Pyton, XML, Netbeans, бібліотеки для роботи з базами знань та JADE для його тестування. Ми обрали JADE серед інших варіантів, тому що він найбільш добре задокументований і є одним з

найбільш зрілих, активних і оновлених фреймворків. Крім того, JADE – це повнофункціональний розширюваний фреймворк на основі Java, який дає достатньо свободи для адаптації його за допомогою розпізнавання тексту та веб-сервісів, щоб зробити наш інструмент розширюваним. Нарешті, JADE є одним з найбільш поширених і активних фреймворків MAS з відкритим вихідним кодом, що робить його надійним фреймворком. Про це є кілька комерційних книг і кілька навчальних посібників в Інтернеті, які роблять його простим у використанні та швидким у вивченні.

Було створено графічний інтерфейс користувача, куди можна завантажувати базу даних знань про домени, характеристики проблеми, функції метамоделей та мета-моделі. У ньому можна виконати або вибрати вручну мікромасив, що характеризується AS, розглядаючи AS-мікромасив для відбору найбільш перспективних кандидатів у мета-моделі. Таким чином запускається MAS, де кожен кандидат у мета-моделі береться агентом, який керує ним для створення групи рішень, а після періоду взаємодії – для пошуку найбільш перспективної групи мета-моделей в якості рішення. Програмний застосунок складається з декількох модулів, як видно на діаграмі UML (Рисунок 3.3.1).

Діаграма UML, відображена на Рисуноку 3.3.1, представляє різні модулі системи. Як ми можемо бачити, модуль Tool є відправною точкою, з якої у нас є два основні шляхи: створити KB за допомогою модуля KB Creator або запустити процес інструменту за допомогою Matching Engine Tool.

KB Creator Module дозволяє нам редагувати та створювати дані про проблеми, характеристики, мета-моделі, можливості та домени, а також значення, які пов'язують кожну характеристику/особливість з доменом. Модуль KB Loader дозволяє завантажувати в Matching Engine Tool такі створені KB. Інструмент Matching Engine Tool складається з підмодулів:

– Meta-model Candidate Chooser (Мета-модель Вибір кандидатів). Це дозволяє вибирати кандидатів на мета-модель з бази знань, щоб використовувати їх у відповідному движку.

– Matching Engine Viewer (Двигун переглядача входжень) дозволяє бачити продуктивність Matching Engine (двигуна входжень) на кожному етапі переговорів і до кінця процесу показувати остаточний вибір групи.

– Domain Selector (Вибір домену). Він вибирає проблемний домен і генерує AS-мікромасив.

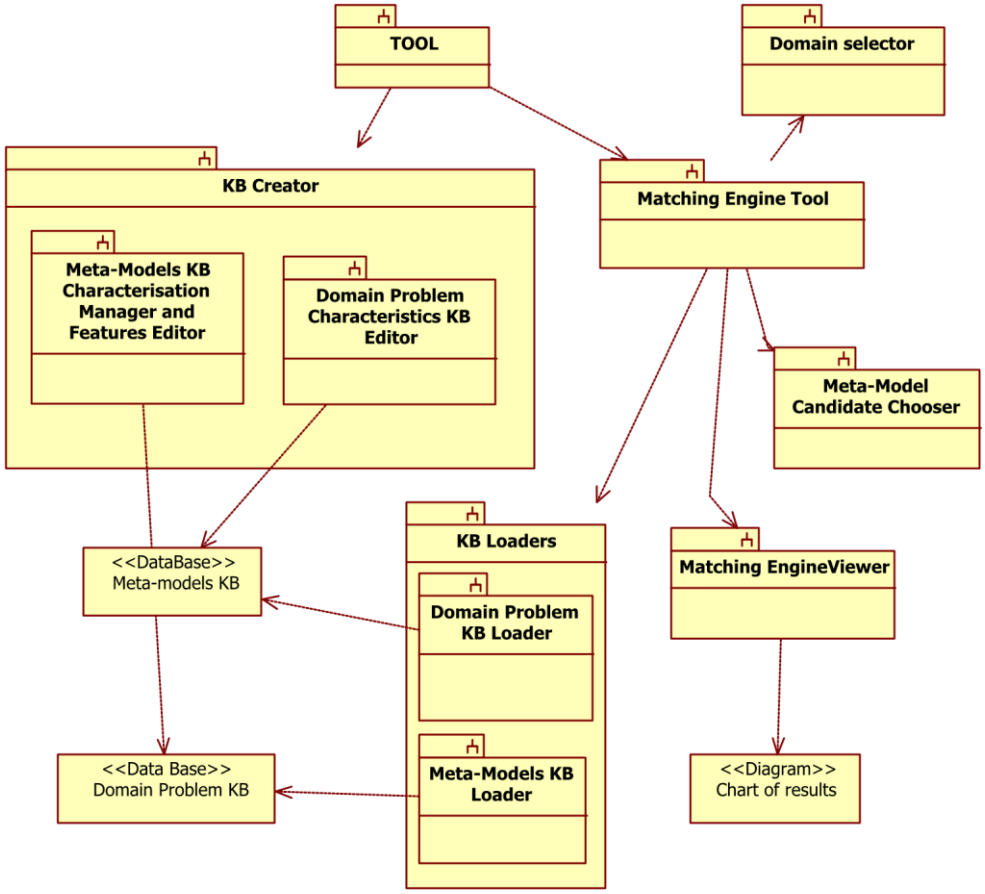


Рисунок 3.3.1 – Діаграма модулів ПЗ

У межах виконаного дослідження нами було розроблено та формалізовано схему основної діяльності програмного забезпечення прототипу, яка наочно представлена на рисунку 3.3.2. Зазначена схема відображає загальну логіку функціонування системи та послідовність ключових етапів обробки даних, починаючи від отримання вхідної інформації й завершуючи формуванням проєктного рішення.

Основна діаграма діяльності вибудувана за сценарієм, який передбачає обов’язкове проходження етапу характеристики специфікації програмного застосунку. На цьому етапі здійснюється аналіз вхідного опису, ідентифікація ключових характеристик проблеми та визначення відповідної предметної (доменної) області. Така характеристика є необхідною передумовою для подальшої обробки, оскільки дозволяє зменшити рівень початкової невизначеності та сформувати структуроване подання вимог застосунку.

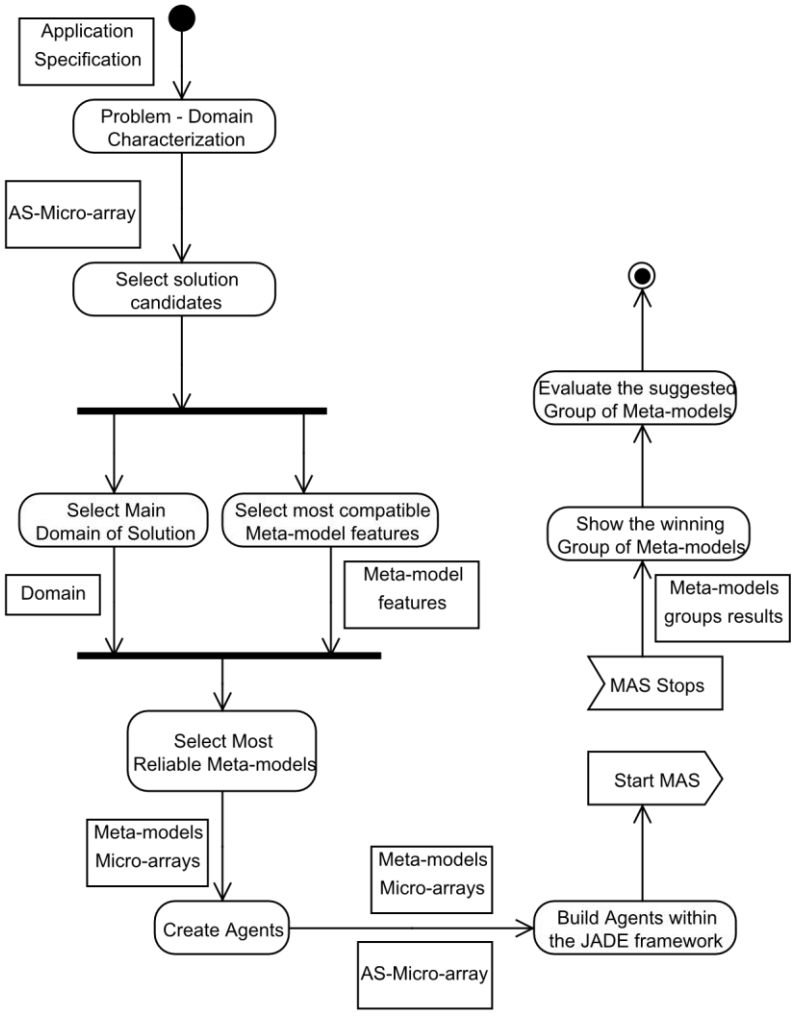


Figure 7.2: Main Activity Diagram

Рисунок 3.3.3 – Діаграма активності когнітивної поведінки агента

Після завершення етапу характеристики система переходить до фази вибору, у межах якої сформований AS-мікромасив використовується як вхідний набір даних для подальшого аналізу. На цьому кроці здійснюється визначення домену або набору доменів рішення, а також виконується процес зіставлення характеристик

проблеми зі всією сукупністю ознак доступних мета-моделей. У результаті такого зіставлення система отримує підґрунтя для обґрунтованого вибору найбільш релевантних мета-моделей і компонентів, які можуть бути використані для побудови рішення на основі агентно-орієнтованої архітектури.

На діаграмі активності, представленій на Рисунку 3.3.3, ми наводимо шлях поведінки, який виконується для кожного агента в рамках БПС. Діаграма поведінки крок за кроком показує, як агент побудував найкращу сумісну групу з урахуванням мета-моделі, яку представляє агент.

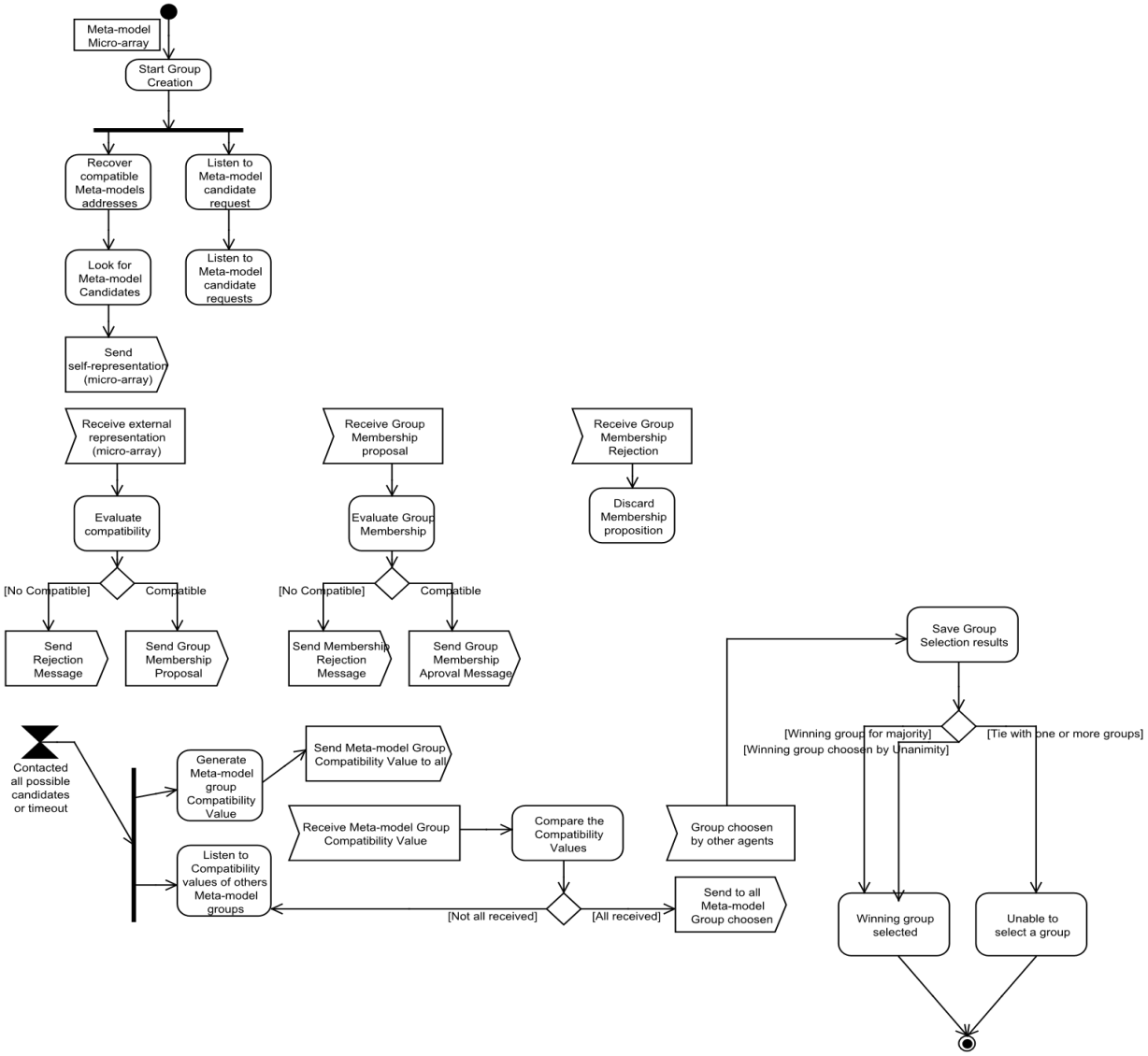


Рисунок 3.3.3 – Діаграма активності когнітивної поведінки агента

Таким чином, запропонована схема основної діяльності прототипу відображає цілісний та послідовний підхід до обробки специфікації застосунку, поєднуючи етапи аналізу, характеристики та вибору в єдиний керований процес, що забезпечує

підтримку прийняття рішень на ранніх стадіях проєктування багатокомпонентних програмних систем.

Мета-аналіз проводиться з отриманням спочатку даних специфікації додатку від користувача. У цьому випадку це файл, який виступає як редактор AS.

1. AS-характеристика. Специфікація програми аналізується, а потім характеризується з метою пошуку характеристик проблеми та предметної області. Це робиться з урахуванням інформації, наданої у файлі. Результатом такого аналізу є AS-Micro-масив.

– Знайти характеристики. Специфікація Додатку аналізується урахуванням Бази знань характеристик, задачі та структур Net. В результаті виходить список характеристик задачі зі значенням відношення.

– Знайти домен. Використовуючи перелік знайдених характеристик задачі, База знань про характеристики проблеми використовується з метою обчислення статистичних значень і визначення області множини характеристик проблеми.

– Побудувати AS-мікро-масив. AS-Micro-масив створюється з урахуванням знайденої доменної області та переліку проблемних характеристик. Він являє собою специфікацію програми.

2. Вибір ММ. Вибір ММ здійснюється з використанням AS-Micro-Array як критеріїв відбору. Підбір здійснюється за допомогою бази знань характеристик ММ.

– Вибір мікроматриці ММ. Процес зіставлення виконується з оцінкою AS-MicroArray з кожним ММ-Micro-Array Бази знань. В кінці в якості кандидатів вибирається набір кращих ММ-мікро-матриць.

– Створення агентів ММ. Агенти будуються з використанням кожного кандидата в ММ-Micro-Array і копії AS-Micro-Array. Кожен агент має дві основні поведінкові завдання: обмін та відбір.

– Груповий обмін. Кожен агент повинен створити групу рішень, домовляючись про членство з іншими агентами. Це робиться з урахуванням лише відмінних від власних типів ММ та оцінки сумісності між ММ-мікро-

масивами. Цей процес завершується, коли всі агенти будуть зв'язані між собою, або коли час виконання закінчиться.

– Вибір групи. Коли кожен агент має групу рішень, результати кожної групи рішень розподіляються між усіма агентами. Кожен агент порівнює сумісність всієї групи рішень з масивом AS-Micro, щоб вибрати найкращий. Нарешті, кожен агент голосує за групу-переможця, відправляючи свій голос спостерігачеві.

3. Показ результатів. Результати відображаються у вікні з урахуванням голосів агентів. Таким чином, виявляється виграшна група рішень. Якщо між двома або більше групами нічия, система не може забезпечити групу-переможця.

ММ-характеристика формується на основі вже наявної мета-моделі та передбачає вилучення з неї необхідної інформації для подальшого подання у вигляді ММ-MicroArray. Така характеристика спрямована на формалізований опис властивостей мета-моделі, що робить можливим її порівняння, аналіз і використання в процесі прийняття рішень. У межах цього процесу мета-модель розглядається не як цілісна структура, а як сукупність окремих ознак, які можуть бути зіставлені з вимогами конкретної задачі.

Зокрема, на першому кроці визначаються знайдені особливості мета-моделі, тобто здійснюється її опис через набір характеристичних ознак у межах відповідного класу. Такі ознаки відображають ключові функціональні та концептуальні властивості мета-моделі, які мають значення для побудови рішення. На наступному етапі виконуються оцінені домени, де до кожної функції мета-моделі додаються статистичні значення ефективності або успішності її використання в межах певної області розв’язання задач. Це дозволяє кількісно оцінити, наскільки дана мета-модель є придатною як компонент рішення для конкретного домену та забезпечує обґрунтований вибір серед альтернатив.

Реалізований прототип програмного забезпечення побудований відповідно до навігаційної схеми, поданої на рисунку 3.3.4, яка відображає загальну структуру та логіку взаємодії його складових. Архітектурно прототип складається з трьох

основних частин: центральної, яка є головним вікном системи, та двох допоміжних частин, що виконують роль редакторів баз знань для керування характеристиками.

Основне вікно прототипу забезпечує користувачеві можливість послідовного переходу між ключовими модулями системи, зокрема вікнами AS-характеристики, MM-Candidate Selector та MAS-Observer. Це дозволяє не лише виконувати запропонований підхід крок за кроком, але й здійснювати спостереження за перебігом процесу, аналізувати проміжні результати та контролювати вибір мета-моделей у межах загальної архітектури рішення.

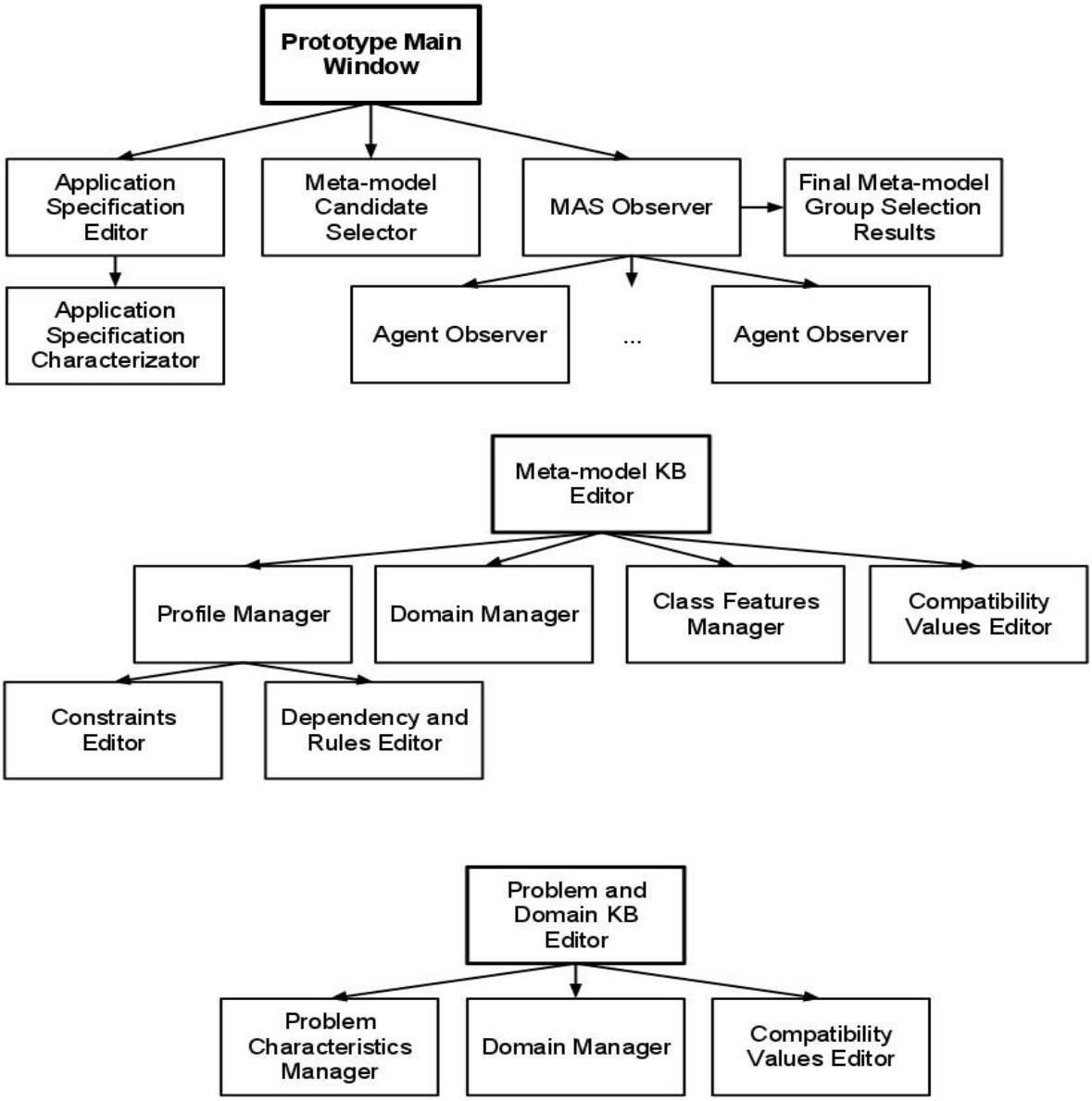


Рисунок 3.3.4 – Діаграма навігації по системі

Діаграми класів, визначені для класів агентів, показані на Рисунку 3.3.5 У наступному розділі ми наводимо частину інструменту.

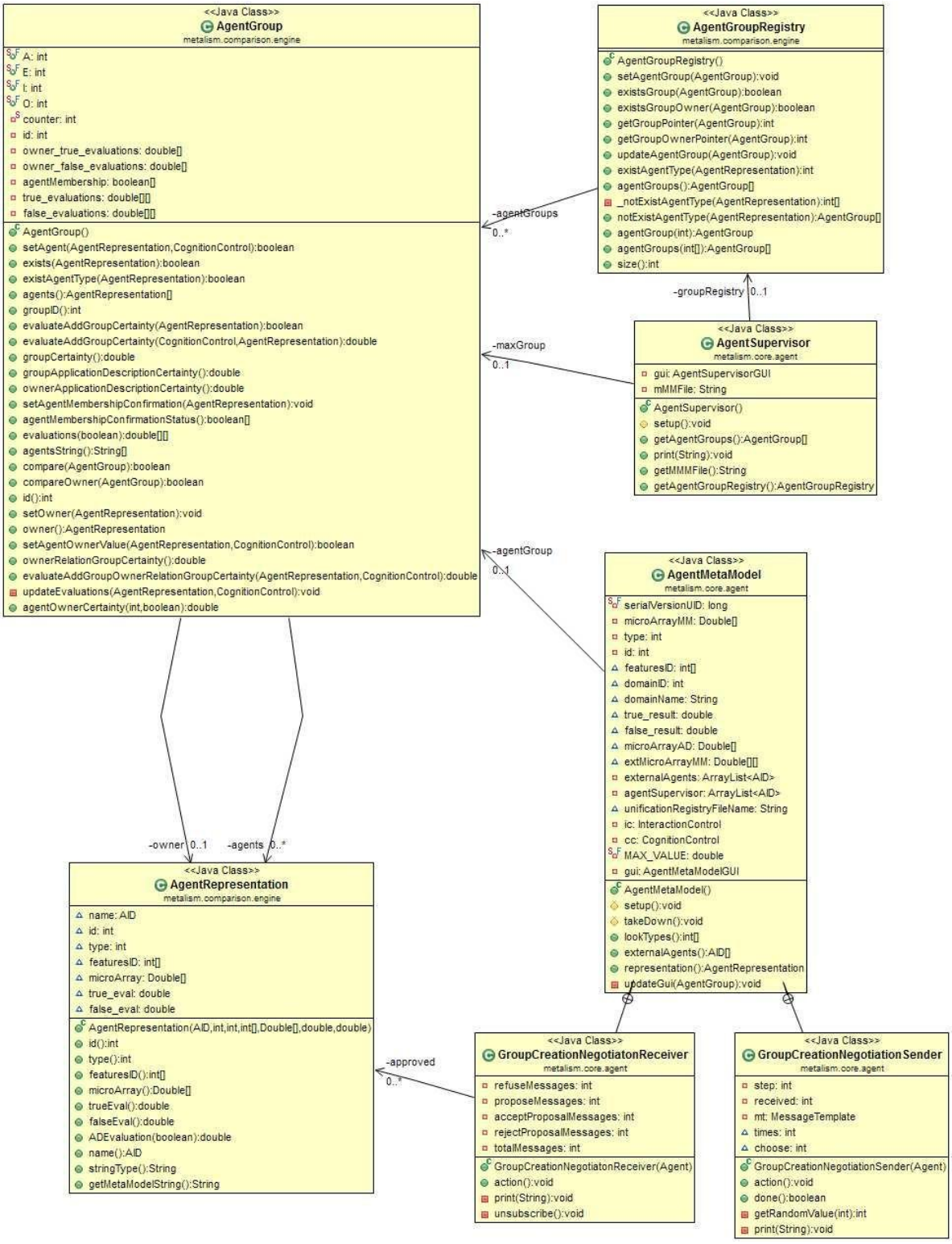


Рисунок 3.3.5 – Діаграма класу агента

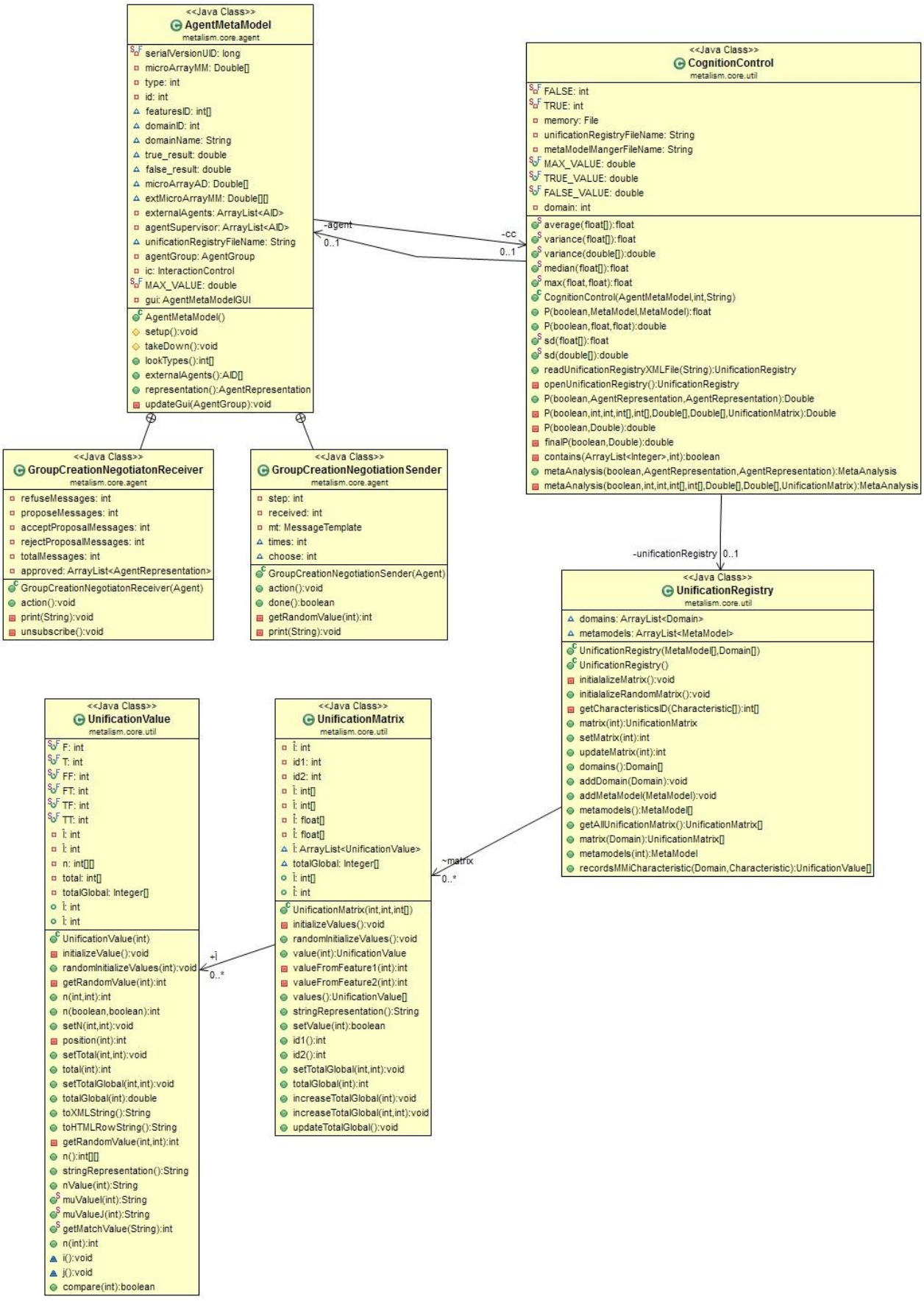


Рисунок 3.3.6 – Діаграма класу розпізнавання агента

3.4. Висновки третього розділу

У третьому розділі магістерської роботи було спроєктовано архітектуру багатоагентної програмної системи та обґрунтовано її модульну реалізацію як інструмента підтримки прийняття рішень на ранніх етапах проєктування БПС. Показано, що хоча процес AS-характеризації не є основним предметом дослідження, для запуску центральної фази механізму узгодження, необхідно визначити мінімально достатній підхід до характеристики текстової специфікації застосування. Запропоновано трактувати таку характеристику як допоміжний «супутниковий» етап, результатом якого є формування машинозчитуваного представлення у вигляді AS-MicroArray, придатного для подальшого зіставлення з характеристиками мета-моделей.

Розроблена архітектура поєднує багатоагентний підхід із механізмами шаблонного аналізу тексту та онтологічно структурованою базою знань. Встановлено, що саме багаторівнева обробка тексту (абзаци → речення → слова) забезпечує формування зважених текстових шаблонів, концептуально подібних до баєсової мережі, та дозволяє працювати з невизначеністю у вхідних описах. Важливим результатом є обґрунтування доцільності накопичення статистичних даних щодо зв’язку між текстовими структурами, проблемними ознаками та доменами, що створює передумови для підвищення точності характеризації та повторного використання результатів аналізу.

У межах розділу визначено склад агентів і розподіл їхніх функцій у процесі AS-характеризації: агент інтерфейсу взаємодіє з користувачем і керує введенням даних; агент характеризації планує та координує етапи аналізу; сервісні агенти забезпечують підключення веб-сервісів (Perl-скрипти, доступ до онтологій); агенти відображення, детектора ознак і локатора доменів виконують когнітивне зіставлення знайдених шаблонів із знаннями, наявними в онтологіях. Така організація підтверджує гнучкість підходу: зміни регулярних виразів і онтологічних записів можуть виконуватися локально (на рівні відповідних агентів) без перероблення всієї

системи, що є практично важливим для адаптації до нових предметних областей і сценаріїв.

Окремо систематизовано логіку роботи механізму добору мета-моделей: від формування AS-мікромасиву та вибору домену до відбору кандидатних MM-MicroArray, створення агентів-кандидатів, переговорного формування груп рішень, оцінювання сумісності й голосування за найкращу групу. Запропоновано також формалізацію MM-характеристики як процесу вилучення ознак мета-моделі та приєднання статистичних значень ефективності в доменах, що дає можливість порівняння і обґрунтованого вибору компонентів рішення.

Реалізовано прототип на основі Java/JADE із застосуванням Python, XML та інтеграцією модулів керування базами знань. Побудовано UML-діаграми модулів і діаграми активності, які підтвердили цілісність сценарію «характеризація → відбір → узгодження → візуалізація результату». У підсумку доведено, що запропонована архітектура створює працездатну основу для зниження початкової невизначеності вибору методологій і мета-моделей, підтримує проєктувальника на стартових кроках розробки БПС та забезпечує масштабованість і розширюваність інструмента завдяки агентній декомпозиції, онтологічним знанням і стандартизованому поданню результатів аналізу.

4 ПРОГРАМА РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ ТА ВАЛІДАЦІЯ ПРОГРАМНОГО ЗАСТОСУНКУ

4.1 Реалізація програмного застосунку. Графічний інтерфейс користувача

Графічний інтерфейс складається з двох різних модулів:

1. Генератор бази знань. Цей модуль має два підмодулі:

– Редактор рішень з предметною зоною. Це дозволяє створювати КВ доменів і проблемних характеристик (Рисунок 4.1.1).

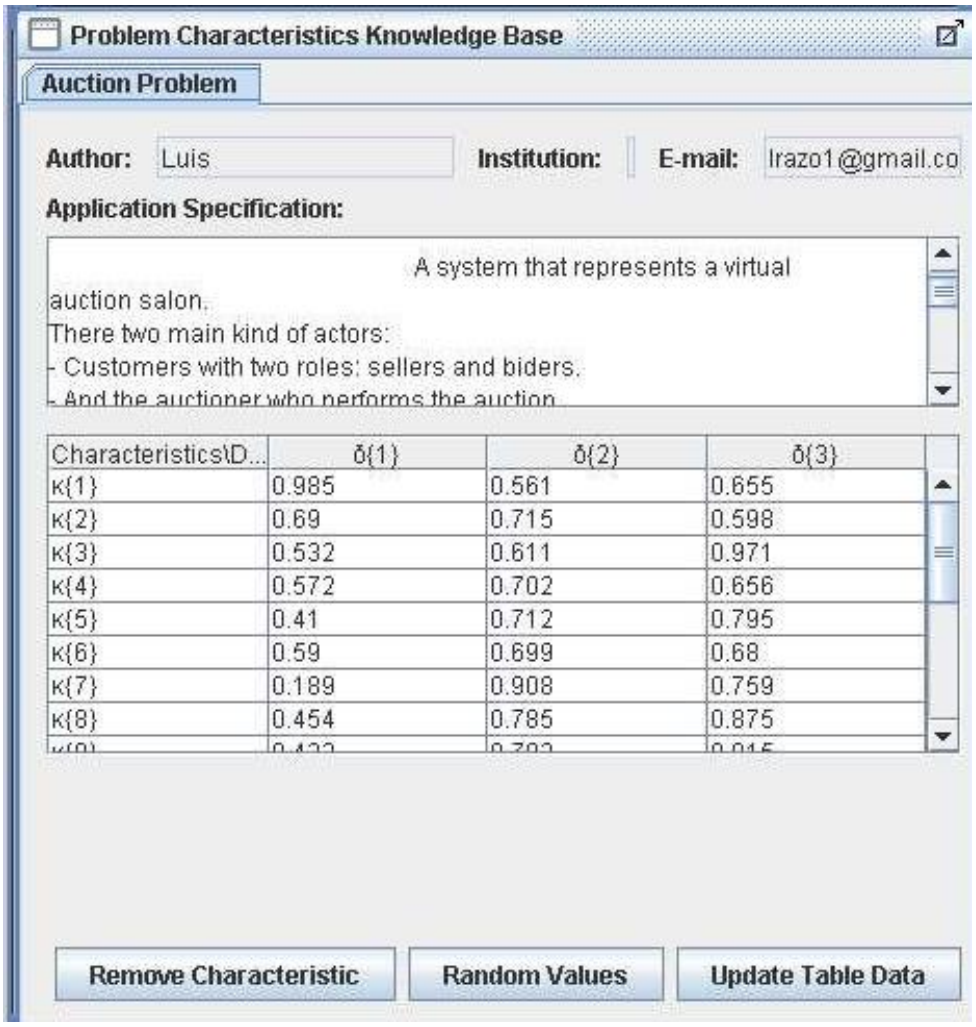


Рисунок 4.1.1 – Головна сторінка редактора рішень

– Менеджер мета-моделі. Це дозволяє визначити особливості метамоделі і, отже, створити мета-модель з використанням цих ознак (Рисунок 4.1.2)

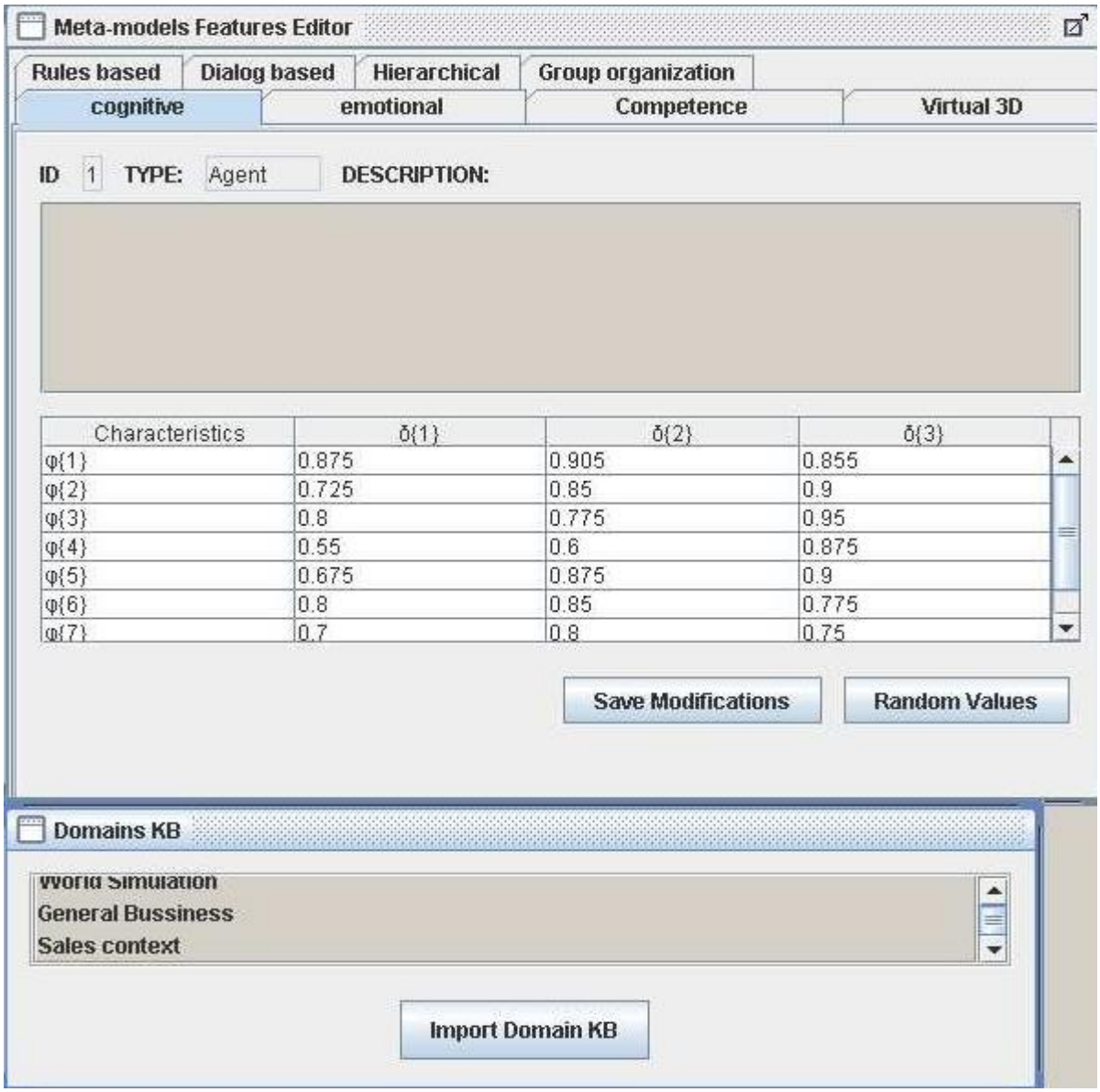


Рисунок 4.1.2 – Вікно функцій мета-моделі

2. Модуль узгодження. Інструмент, що дозволяє виконати процес узгодження (, однак він складається з наступних підмодулів:

– Навантажувачі КБ. Передбачено дві кнопки, за допомогою яких можна завантажити КВ з проблемою домену та базу даних мета-моделі для виконання процесу зіставлення;

– Вибір домену. Він дозволяє вибрати домен відповідно до даних, що містяться в мікромасиві (Рисунок 4.1.3). Це можна вибрати вручну або отримати за допомогою іншого підходу;

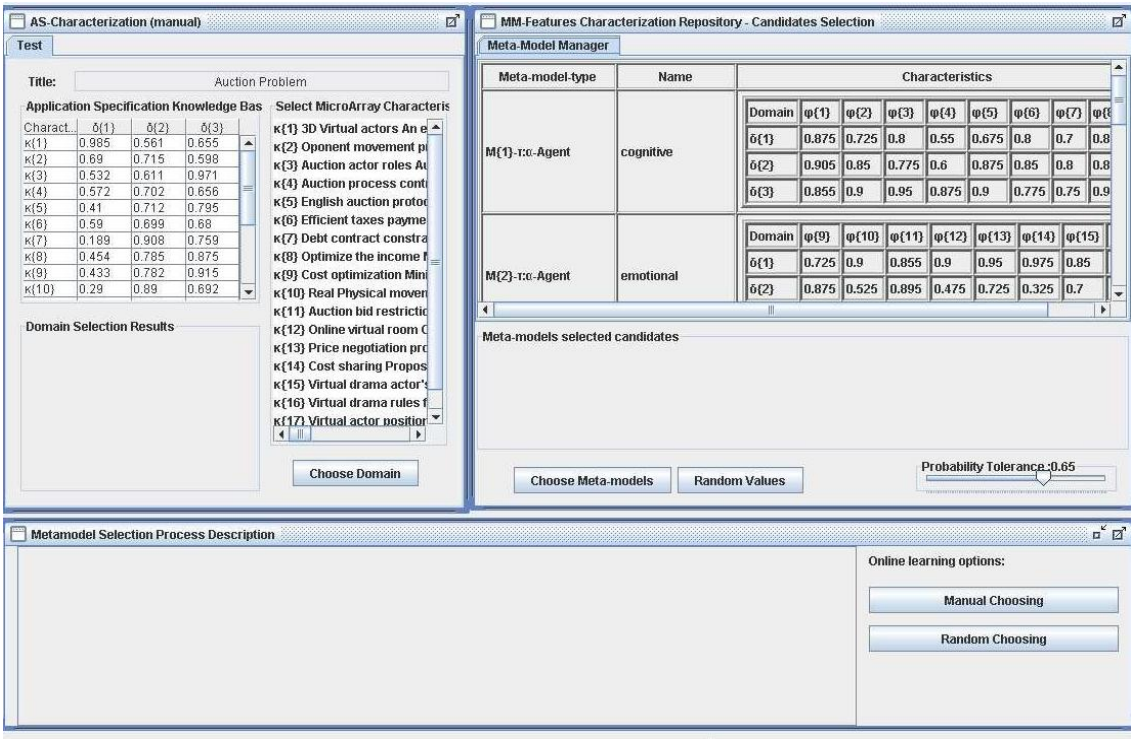


Рисунок 4.1.3 –Вікно рішень

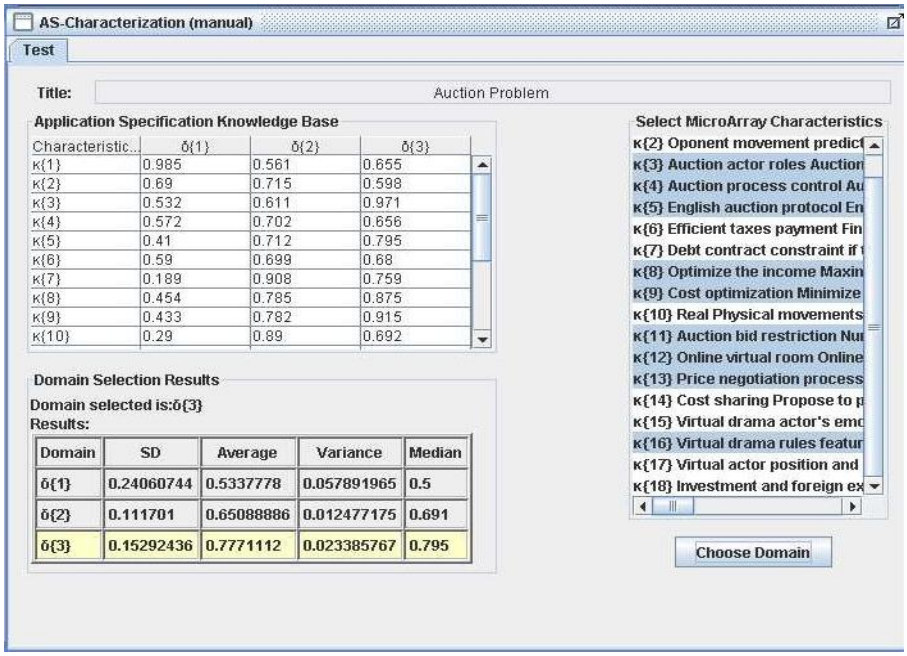


Рисунок 4.1.4 –AS-характеристики

– Мета-модель вибору кандидатів. Використовуючи дані КВ мета-моделей, обраний домен та мікромасив як критерії відбору, цей модуль дозволяє вибрати кандидатську підмножину мета-моделей з КБ (Рисунок 4.1.4);

– Відповідний переглядач. Коли кандидати на мета-модель обрані та натиснута кнопка запуску двигуна, запускається відповідний двигун, а потім модуль відповідного перегляду відображає поточний стан агентів, які володіють мета-моделями MAS. Він показує дані агента про власність мета-моделі та поточну створену групу рішень. В кінці процесу можливе відображення діаграми, що показує результати роботи груп, щоб наочно виділити кращу групу з урахуванням ймовірності успіху (Рисунок 4.1. 5);

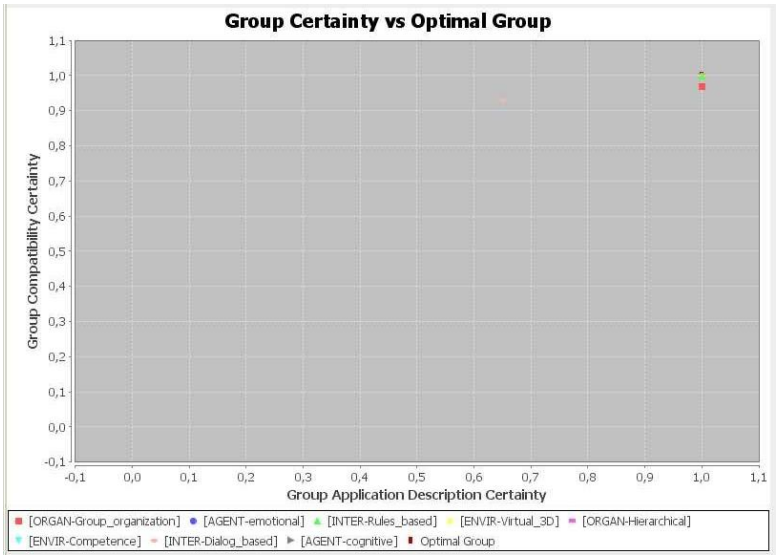


Рисунок 4.1. 5 – Вікно результати роботи груп

Бази знань можуть бути створені вручну, або за допомогою автоматизованого підходу, шляхом генерації з існуючих, для створення доменів, AS-характеристик і ММ-функцій КВ за допомогою веб-аналізу або аналізу документів. Як зазначалось в попередньому розділі, ми вважаємо це питанням, яке не є основним акцентом даної дипломної роботи. Отже, ми впровадили ці програмні модулі для створення таких КВ вручну.

Ми можемо додати характеристики і домени, для кожного відношення «як-характеристика – домен» ми повинні визначити значення. Однак, коли ми додаємо новий домен, або нові AS-характеристики. Значення за замовчуванням дорівнює 0.5, що означає максимальну невизначеність або ентропію.

Також ми можемо редагувати та створювати мета-моделі та їх особливості.

Крім того, для виконання задачі на входження ми створили модуль Matching Engine, графічний інтерфейс якого показаний на рисунку 4.1.1, де ми завантажуємо КВ, створену за допомогою редактора

Графічний інтерфейс програмного застосунку поділяється на три частини:

- AS-характеристика;
- характеристика ММ-особливостей;
- процес вибору мета-моделі.

Далі візуалізуємо AS-Характеристики КВ і можемо вручну виконувати характеристику, вибираючи набір доступних характеристик, таким чином з урахуванням такого набору автоматично вибираємо домен приналежності AS (Рисунок 4.1.4). Однак ми можемо завантажити автоматично охарактеризовані значення з раніше охарактеризованого файлу мікромасиву.

Процес відбору детально описаний у попередньоному розділі. На рисунку 4.1.6 представлені результати прототипу такого процесу відбору.

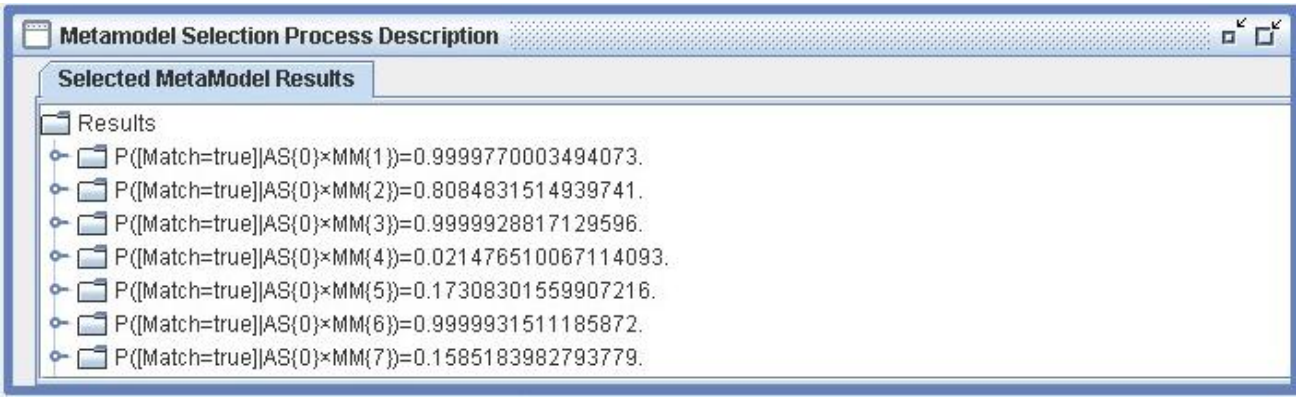


Рисунок 4.1.6 – Вікно ММ-функцій

Як зазначалость нами раніше, пропонується використовувати суспільство агентів, які приймають рішення індивідуально, а також колективно. Таким чином, на рисунку 4.1.6. ми бачимо спостерігачі MAS, які показують внутрішній стан кожного агента під час їх взаємодії між собою для побудови груп рішень. Внутрішній процес такого агента детально описаний раніш, а приклад показаний на рисунку 4.1.7.

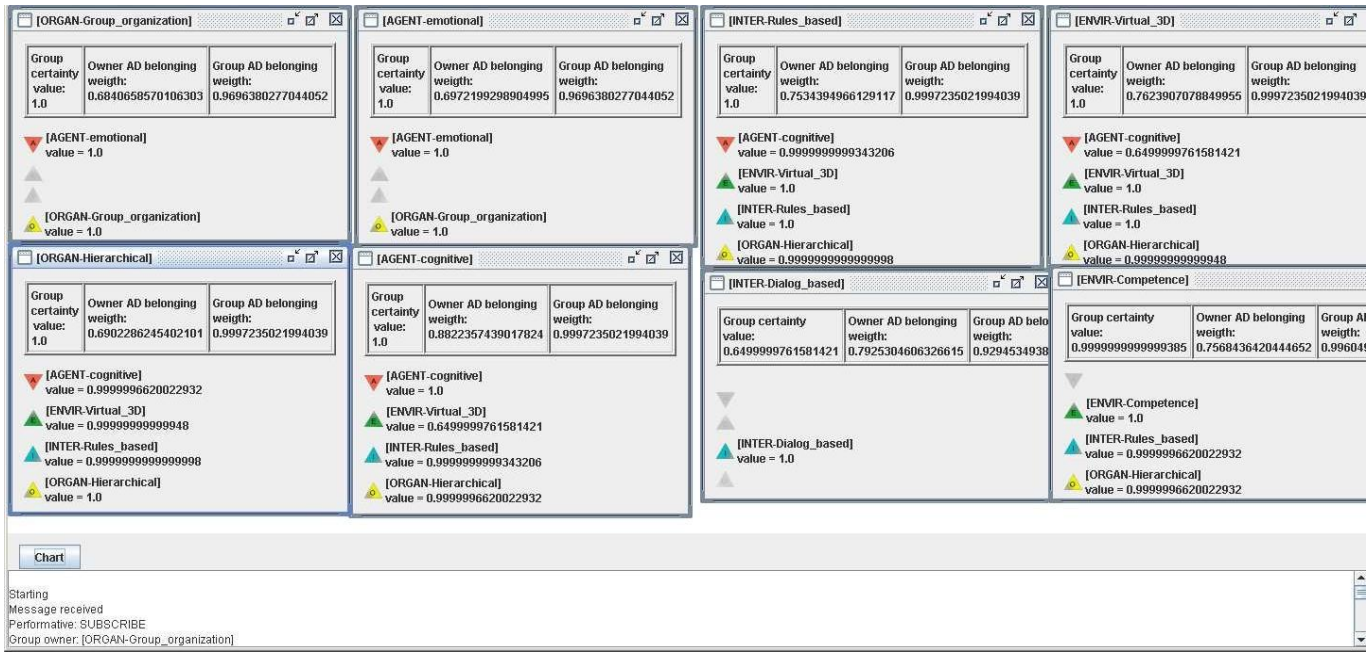
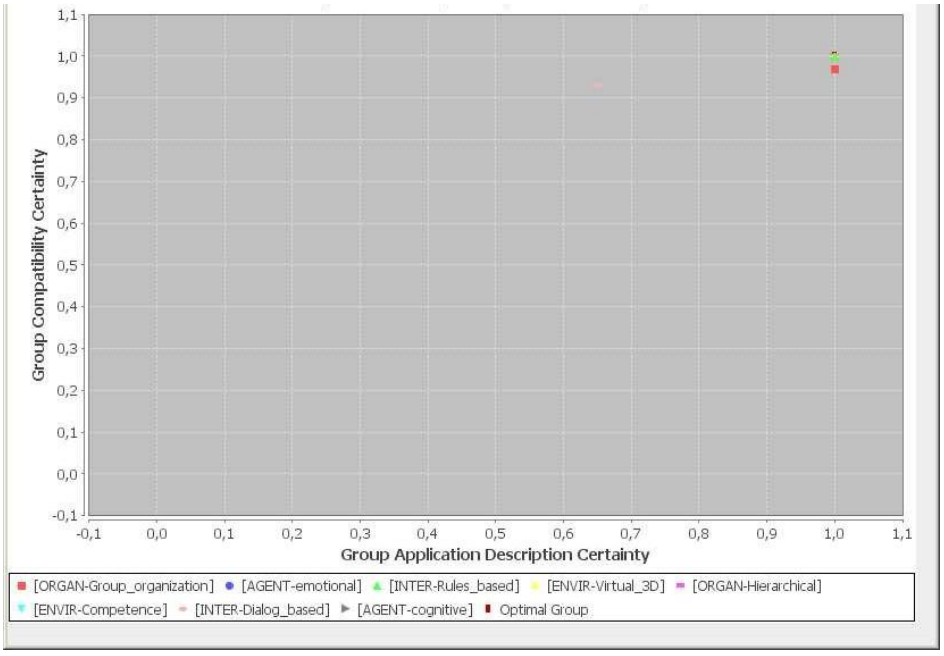


Рисунок 4.1. 7 – Діаграма Observers

Далі ми можемо спостерігати за груповою діаграмою з урахуванням результатів роботи агента. Ми можемо бачити оптимальні або очікувані значення груп на тлі отриманих груп. Ми розглянемо дві характеристики: впевненість щодо специфікації застосунку (опису) та впевненість сумісності груп.

Найбільша ймовірність по двох осях наближаться до 1,0, найнижча ймовірність до 0,0 (Рисунок 4.1.8).



4.1.8 – Результат очікуваних значень

4.2 Тестування та валідація. Приклад характеристики AS

Щоб протестувати наш метод ми створили кейс-стаді для його перевірки. Поряд з кожним етапом запропонованого методу наведемо приклади, які демонструють роботу нашого рішення на різних етапах.

Ми починаємо тестування з прикладу етапу характеристик специфікації додатку (AS Характеристики). У цьому кейсі проаналізовано систему, яка вимагає створення віртуальної тогової платформи типу Amazon та схожих до нього.

У системі введе основні критерії що відносятьсядо віртуальної тогової платформи. Загалом введемо два основних типи акторів:

– клієнтів з двома типами ролей, продавців та учасники торгів;

– ініцитатор, який проводить торгівлю за зазначеними правилами.

Як клієнти, покупці учасники площадки, можуть створити свого представника, який налаштовує поведінку на свій розсуд. В нашому випадку це віртуальну сутність, яка представляє покупця на площадці. Ці віртуальні сутності мають настроювану поведінку, тобто клієнт може додати такі функції:

– максимальний бюджет на список або категорію товарів.

– вони можуть приймати правила робити, або не робити ставки.

З іншого боку, віртуальною площадкою керуватиме автоматизована віртуальна сутність, яка створюватиме лоти, приймаючи запропоновані товари для продажу від клієнтів, які хочуть продати на аукціоні. Таким чином, ми будемо приймати товари з початковою вартістю і бажаною продажною вартістю, тому до кінцевої вартості реалізації ми додамо 25% витрат на різні збори та податки. Кожна площадка може бути налаштована таким чином, що клієнт-продавець може вибрати між чотирма основними типами правил торгової площадки:

– українська площадка;

– китайська площадка;

– американська площадка;

– європейська площадка.

Як правило, обидва домени онтології бази знань – C_{KB} і D_{KB} – використовуються в процесі AS-характеристики, мають велику кількість AS-характеристик і доменів відповідно. Однак з міркувань простору число AS-характеристик і доменів, що використовуються на наступних етапах прикладу нами було скорочено, і вони не повними.

Використовуваний підхід процесу AS-характеристики базується на текстових AS-характеристиках. Він використовує розпізнавання образів на основі тексту - подібне до того, що використовується в статистичному машинному перекладі і використовує регулярні вирази. Однак процес характеризації AS може бути реалізований по-іншому, використовуючи інші підходи, проте такі процеси характеризації не є метою даної роботи. Таким чином, КВ доменів і використовуваних характеристик виглядає як показано в таблицях 4.2.1- 4.2.3.

Таблиця 4.2.1 – Домен D_{KB}

Домен	Опис
δ1	Домен моделювання.
δ2	Домен на основі переговорів.
δ3	Домен контексту публічних продажів.

Таблиця 4.2.2 – Домен C_{KB}

Ознака	Опис
к1	Середовище з віртуальними акторами .
к2	Аналіз опонента і прогнозування його рішень.
к3	Актори аукціону – клієнти з ролями покупця і продавця.
к4	Суб'єкт аукціону контролю процесу аукціону.
к5	Протокол аукціону.
к6	Найкращі підказки для зменшення сплати податків

к7	У разі несплати боржником боргу згідно з договором виникає право на заставу.
к8	Максимізувати дохід або прибуток.
к9	Мінімізувати собівартість або ціну.
к10	Рухи на аукціоні.
к11	Обмеження кількості ставок.
к12	Віртуальна кімната в режимі он-лайн
к13	Процес узгодження ціни.
к14	Пропозиція оплачувати витрати та ділитися порівну.
к15	Актори виражають додаткові побажання
к16	Визначені правила для акторів
к17	Характеристи акторів на досвіді.
к18	Курси валют для олати.

Таблиця 4.2.3 – AS_{KB}-характеристик

	Характеристики								
Домен	к1	к2	к3	к4	к5	к6	к7	к8	к9
δ1	0,975	0,68	0,522	0,562	0,4	0,58	0,179	0,444	0,423
δ2	0,551	0,705	0,601	0,701	0,702	0,689	0,907	0,775	0,772
δ3	0,645	0,588	0,961	0,656	0,785	0,67	0,749	0,865	0,905
Домен	к10	к11	к12	к13	к14	к15	к16	к17	к18
δ1	0,804	0,28	0,883	0,109	0,2	0,934	0,881	0,854	0,38
δ2	0,301	0,88	0,47	0,681	0,844	0,301	0,585	0,513	0,813
δ3	0,184	0,682	0,685	0,888	0,866	0,379	0,679	0,36	0,701

У процесі характеризації вибираються тільки ті ознаки, які виявляються в рамках аналізованої AS, а потім, коли процес характеризації закінчився, ми маємо результат наведений у таблицях 4.2.4 та 4.2.5.

Таблиця 4.2.4 – AS_{AAS}-характеристики

	Отримані характеристики								
Домен	к3	к4	к5	к8	к9	к11	к12	к13	к16
δ1	0,522	0,562	0,401	0,444	0,423	0,280	0,883	0,109	0,881
δ2	0,601	0,701	0,702	0,775	0,772	0,880	0,470	0,681	0,585
δ3	0,961	0,646	0,785	0,865	0,905	0,682	0,685	0,888	0,679

Таблиця 4.2.5 – AS_{AAS}-характеристик

	Середнє значення	Стандартне відхилення	Дисперсія	Медіана
δ1	0,510	0,254	0,065	0,454
δ2	0,694	0,121	0,015	0,702
δ3	0,798	0,119	0,014	0,795

На графіку (Рисунок 4.2.1) – показані AS-характеристики зі значеннями доменів їх приналежності, скоригованими. Характерна тенденція показує, що областю з найбільшою правдоподібністю є δ3, оскільки всі значення їхніх характеристик знаходяться в області найбільшої правдоподібності. Крім того, δ3 має найнижчу дисперсію та стандартне відхилення, однак медіана δ3 є найвищою серед інших доменів. Отже, це означає, що AS-характеристика зменшує невизначеність щодо приналежності домену AS через локалізацію характеристик. Таким чином, кінцевий результат виглядає як показано у Таблиці 4.2.6

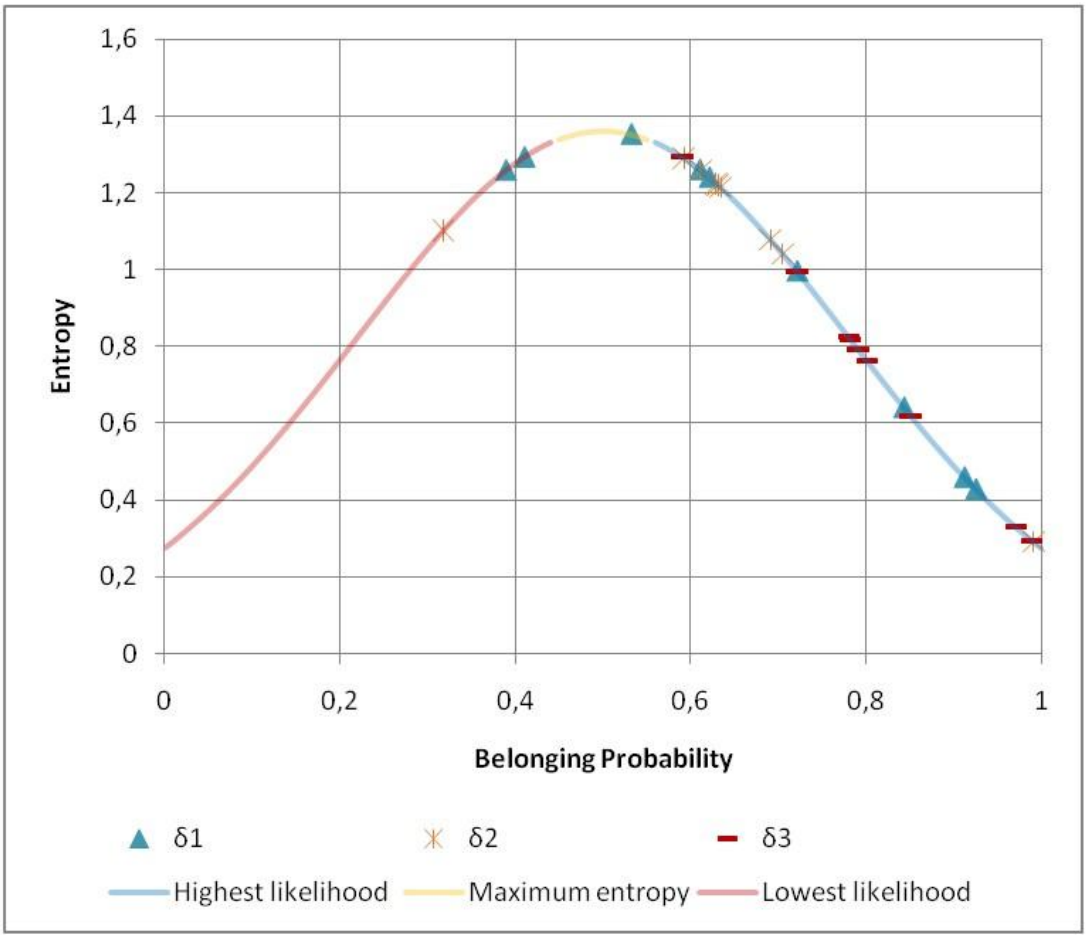


Рисунок 4.2.1 – Діаграма AS характеристик площадок

Таблиця 4.2.6 – AS_{AAS}-характеристик

	Вибрані характеристики								
Домен	к3	к4	к5	к8	к9	к11	к12	к13	к16
δ3	0,971	0,656	0,795	0,875	0,915	0,692	0,695	0,898	0,689

У цьому основному випадку, коли використовується тільки три домени, ми вибираємо тільки один найбільш адаптований, однак відмітимо, що можна вибрати більше одного домену.

4.3 Тестування та валідація. Приклад ММ-характеристики

У контексті прикладу характеристики *ASaas*, розглянутого в пункті 4.2 та розглядаючи одні й ті ж домени δ1, δ2 і δ3, ми використовуємо наступні ММ-типи (Таблиця 4.2.7):

Таблиця 4.2.7 – ММ-типи

Назва	Типи	
Агентів	α_1– Когнітивний агент	α_2 – Емоційний агент
Середовища	ε_1 – Компетентне середовище	ε_2 – Віртуальне середовище
Взаємодії	ι_1 – Базовані на основі правил	ι_2 – Діалоговобазовані
Організацій	o_1 – Ієрархічна організація	o_2 – Групова організація

Для того, щоб отримати всі значення характеристик мета-моделей та додати їх в КВ, спочатку були проаналізовані ММ та з кожної ММ були абстраговані ММ-ознаки і визначені за допомогою текстових пропозицій (Таблиця 4.2.8). Відсоткові значення ефективності були отримані з урахуванням інформації, отримані з дослідження наукових статей (Таблиця 4.2.9). Однак цей процес можна автоматизувати за допомогою підходу інженерії знань та розпізнавання образів, але цей процес виходить за рамки цієї дисертації. Для когнітивного агента та розподіленого напівспостережуваного середовища ознаки визначаються за допомогою позначень, для агента та віртуального середовища – ознаки, описані в 3 розділі. Ознаки взаємодії, засновані на правилах, а функції взаємодії на основі діалогів були абстраговані.

Тому ми використовуємо значення ефективності ММ КВ, як видно з таблиці 4.2.8. Таким чином визначаючи разом обидва КВ: ММ та значення ефективності.

Таблиця 4.2.8 – Значення ефективності ММ КВ

ММ	Значення ефективності	ММ	Значення ефективності
Ф1	Пізнання	Ф33	Симуляція передумов та пост-умов
Ф2	Уявлення про себе та інших	Ф34	Мова
Ф3	Стратегічна компетентність	Ф35	Комунікація
Ф4	Незалежна продуктивність	Ф36	Тайм-аути
Ф5	Поведінка знання про організацію	Ф37	Розділені повідомлення

Ф6	Соціальне слухання інших	Ф38	Трансляція
Ф7	Тактика дії та рішення	Ф39	Комунікація
Ф8	Репрезентація третіх агентів	Ф40	Повідомлення
Ф9	Емоційне пізнання	Ф41	Зустріч
Ф10	Поведінка на основі мети	Ф42	Повідомлення без відповіді
Ф11	Агентська взаємодія через датчики	Ф43	Штраф
Ф12	Агентська взаємодія з середовищем	Ф44	Відповідь-Відповідь
Ф13	Драматургія	Ф45	Передача
Ф14	Навичка торгу	Ф46	Мінімальний набір слів
Ф15	Навичка на основі діалогу	Ф47	Діалог
Ф16	Динамічна генерація цілей	Ф48	Тайм-аут
Ф17	Навичка частково спостережувана	Ф49	Кілька слухачів
Ф18	Детерміновані послідовні дії	Ф50	Діалог
Ф19	Динамічні зміни	Ф51	Ідея
Ф20	Безперервна часова лінія	Ф52	Шторм
Ф21	Мультиагентна підтримка	Ф53	Вид
Ф22	Конкурентні правила	Ф54	Діалог
Ф23	Багатолокаційний епізодичний порт	Ф55	Спостережуваний діалог
Ф24	Фізична взаємодія	Ф56	Секретний діалог
Ф25	Симуляція передумов та пост-умов	Ф57	Тайм-аут
Ф26	Правила членства	Ф58	Емоційний контекст
Ф27	Група	Ф59	Дозволені дії
Ф28	Оцінка групи	Ф60	Ролі
Ф29	Група знань	Ф61	Рівень оновлення
Ф30	Обмеження групи	Ф62	Дозволи на зв'язок
Ф31	Протокол членства	Ф63	Постачальник послуг
Ф32	Ролі учасників групи	Ф64	Довідник

Таблиця 4.2.9 – Значення ефективності ММ-ознак КБ вибірки

Домени	Функції							
$\tau_1 : \alpha 1$	$\phi 1$	$\phi 2$	$\phi 3$	$\phi 4$	$\phi 5$	$\phi 6$	$\phi 7$	$\phi 8$
$\delta 1$	0,875	0,725	0,8	0,55	0,675	0,8	0,7	0,825
$\delta 2$	0,905	0,85	0,775	0,6	0,875	0,85	0,8	0,85
$\delta 3$	0,855	0,9	0,95	0,875	0,9	0,775	0,75	0,9
$\tau_2 : \alpha 2$	$\phi 9$	$\phi 10$	$\phi 11$	$\phi 12$	$\phi 13$	$\phi 14$	$\phi 15$	$\phi 16$
$\delta 1$	0,725	0,9	0,855	0,9	0,95	0,975	0,85	0,85
$\delta 2$	0,875	0,525	0,895	0,475	0,725	0,325	0,7	0,8
$\delta 3$	0,695	0,4	0,7	0,525	0,625	0,375	0,725	0,825
$\tau_3 : \varepsilon 1$	$\phi 17$	$\phi 18$	$\phi 19$	$\phi 20$	$\phi 21$	$\phi 22$	$\phi 23$	$\phi 24$
$\delta 1$	0,9	0,625	0,875	0,9	0,35	0,95	0,9	0,725
$\delta 2$	0,6	0,7	0,8	0,925	0,8	0,975	0,875	0,6
$\delta 3$	0,725	0,75	0,875	0,975	0,9	0,975	0,9	0,75
$\tau_4 : \varepsilon 2$	$\phi 25$	$\phi 26$	$\phi 27$	$\phi 28$	$\phi 29$	$\phi 30$	$\phi 31$	$\phi 32$
$\delta 1$	0,785	0,725	0,92	0,935	0,965	0,94	0,745	0,891

δ2	0,685	0,725	0,824	0,451	0,395	0,855	0,815	0,785
δ3	0,888	0,795	0,745	0,325	0,47	0,835	0,758	0,698
τ_5 : ι1	ϕ33	ϕ34	ϕ35	ϕ36	ϕ37	ϕ38	ϕ39	ϕ40
δ1	0,95	0,745	0,565	0,855	0,69	0,905	0,72	0,9
δ2	0,885	0,955	0,435	0,745	0,895	0,645	0,91	0,89
δ3	0,89	0,905	0,395	0,745	0,784	0,705	0,89	0,74
τ_6 : ι2	ϕ41	ϕ42	ϕ43	ϕ44	ϕ45	ϕ46	ϕ47	ϕ48
δ1	0,72	0,812	0,86	0,88	0,86	0,91	0,78	0,61
δ2	0,92	0,95	0,901	0,25	0,56	0,89	0,77	0,7
δ3	0,75	0,925	0,68	0,45	0,61	0,78	0,79	0,65
τ_7 : *O1*	ϕ49	ϕ50	ϕ51	ϕ52	ϕ53	ϕ54	ϕ55	ϕ56
δ1	0,905	0,75	0,895	0,9	0,901	0,934	0,889	0,65
δ2	0,815	0,89	0,91	0,92	0,8	0,812	0,875	0,7
δ3	0,885	0,405	0,7	0,651	0,59	0,671	0,7	0,9
τ_8 : *O2*	ϕ57	ϕ58	ϕ59	ϕ60	ϕ61	ϕ62	ϕ63	ϕ64
δ1	0,815	0,79	0,49	0,855	0,789	0,72	0,9	0,888
δ2	0,545	0,651	0,75	0,6	0,645	0,89	0,815	0,955
δ3	0,628	0,455	0,65	0,595	0,785	0,5	0,566	0,85

Беручи до уваги результати, отримані в результаті 20 опитувань, в яких збігів було 15, а неспівставлених – 5, ми маємо наступний фрагмент гістограми (Таблиця 4.2.10):

Таблиця 4.2.10 –Гістограма

Гістограма (δ3)			
j	λj = (θ *i*,μ_k)	N^J_F	N^T_J
129	(θ *3*,μ_1)	0	9
193	(θ *4*,μ_1)	5	14
257	(θ *5*,μ_1)	4	15
449	(θ *8*,μ_1)	5	14
513	(θ *9*,μ_1)	5	0
641	(θ11,μ 1)	0	12
705	(θ12,μ 1)	0	12
833	(θ13,μ 1)	5	14
961	(θ16,μ 1)	5	13

...	...	...	...

Таким чином, отримуємо наступні результати оцінювання (Таблиця 4.10 (фрагмент)):

Таблиця 4.10 – Результати оцінювання

$P([\ Math = True\ ] \mid (\theta i, \mu_к)\ )$									
$(\theta i, \mu_к)$	$\theta 3 = t$	$\theta 4 = f$	$\theta 5 = t$	$\theta 8 = t$	$\theta 9 = t$	$\theta 11 = f$	$\theta 12 = f$	$\theta 13 = t$	$\theta 16 = f$
$\mu 1 = t$	0.588	0.882 грн	0.941 грн	0.882 грн	0.059	0.765 грн	0.765 грн	0.882 грн	0.824 грн
...	...	...	...				...	...	...

4.3 Аналіз результатів

Отже, відповідаючи на запитання (використовуючи вирази з розділу 6.5.3.2):

$$P(\ [Math = True] \mid \lambda_{129} \wedge ... \wedge \lambda_{968}) = 0.9912$$

Результат вказує на те, що, виходячи з досвіду, закладеного в гістограму, ймовірність сумісності мета-моделі α_1 з AS-характеристикою висока. Таким чином, ця мета-модель вибирається в якості кандидата і створюється агент для представлення мета-моделі.

Таким чином, результат вказує на те, що ймовірність сумісності метамоделей α_1 і ε_1 висока (близька до 1), тоді агент, який представляє α_1, додає до своєї групи представлену мета-модель ε_1.

Тобто, агенти взаємодіють між собою для пошуку кращих партнерів по групі, оцінюючи ймовірність сумісності кожного з можливих варіантів. На заключному етапі агенти вибирали між собою найкращу групу, створену з урахуванням найвищих значень сумісності членів групи і сумісності групи з проблемою.

Нарешті, найбільш сумісною групою, знайденою в нашому прикладі, була група, утворена метамоделями: когнітивний агент, компетентнісне середовище, діалогова взаємодія та ієрархічна організація (Рисунок 4.1).

Аналізуючи продуктивність інструменту, ми залучили профайлер Netbeans, щоб з'ясувати, які операції є найбільш значущими та час, необхідний для їх виконання. Термін «профіль» у контексті розробки означає спосіб аналізу продуктивності програми або отримання всебічного розуміння загальної продуктивності програми. У нашому випадку нас цікавить аналіз часу виконання взаємодій агентів, щоб оцінити загальну продуктивність інструменту.

На рисунку 4.3.1 показані результати загального виконання проекту, де кожен метод оцінюється з урахуванням кількості викликів та часу, необхідного для виконання таких викликів.

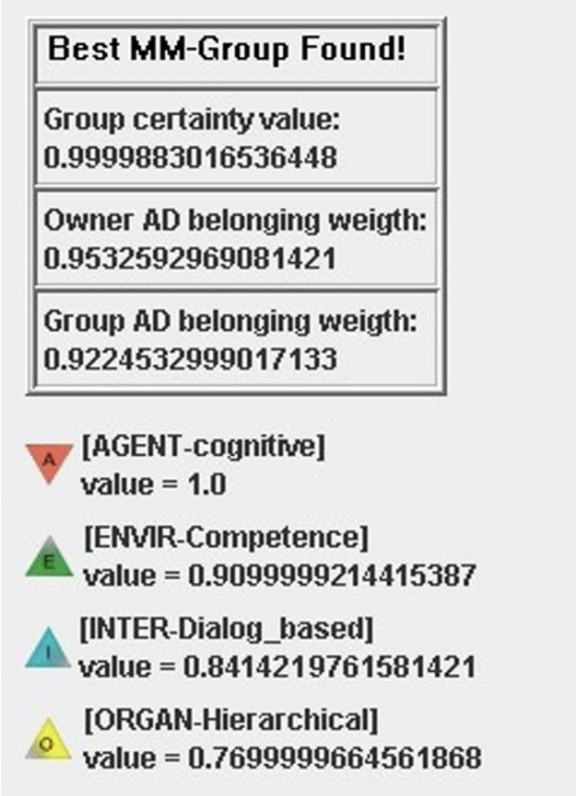
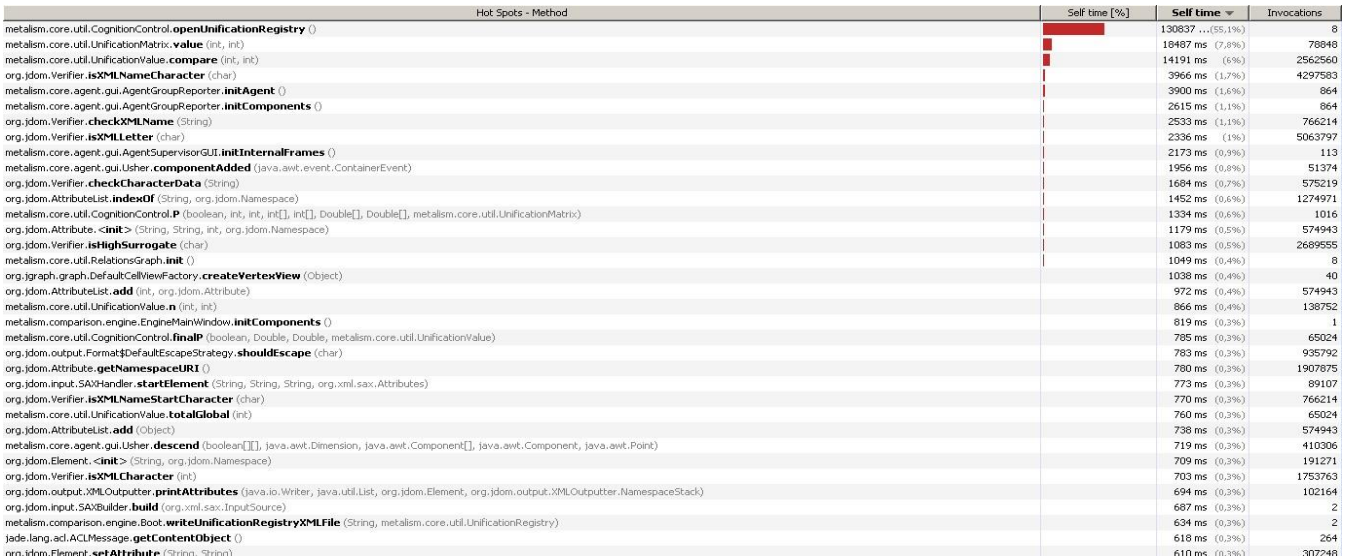


Рисунок 4.3.1 – Скрін результату

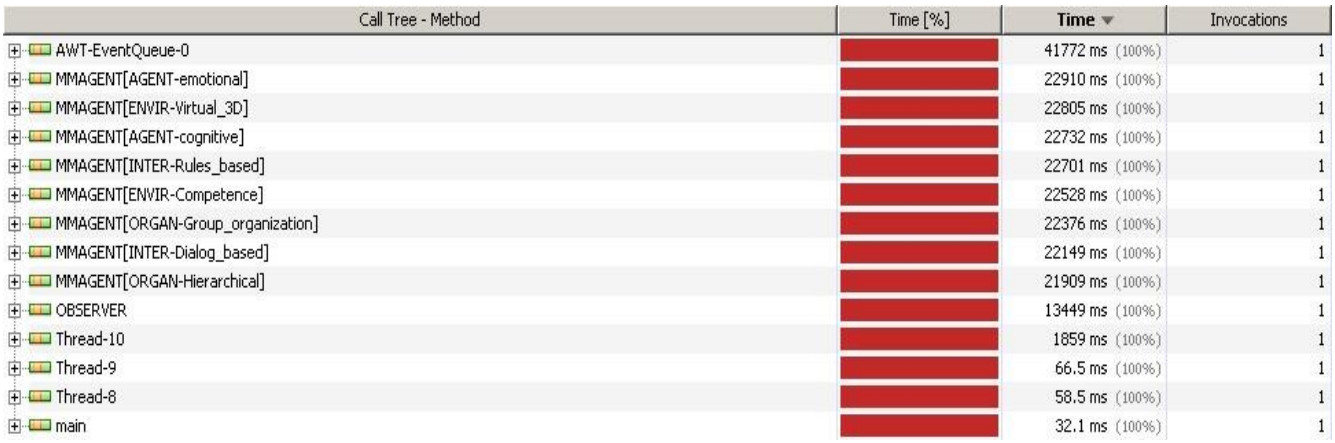
Як видно на малюнку, найбільш затребуваними методами є ті, що пов'язані з читанням/записом реєстру уніфікації. Це означає, що агенти постійно використовують такий реєстр, який зберігає єдину базу знань, щоб оцінити сумісність між метамоделями. Наступний список методів «value» і «compare» пов'язані з такими оцінками сумісності (Рисунок 4.3.2).



Hot Spots - Method	Self time [%]	Self time	Invocations
metalism.core.util.CognitionControl.**openUnificationRegistry** ()		130837 ...(55,1%)	8
metalism.core.util.UnificationMatrix.**value** (int, int)		18487 ms (7,8%)	78848
metalism.core.util.UnificationValue.**compare** (int, int)		14191 ms (6%)	2562560
org.jdom.Verifier.**isXMLNameCharacter** (char)		3966 ms (1,7%)	4297583
metalism.core.agent.gui.AgentGroupReporter.**initAgent** ()		3900 ms (1,6%)	864
metalism.core.agent.gui.AgentGroupReporter.**initComponents** ()		2615 ms (1,1%)	864
org.jdom.Verifier.**checkXMLName** (String)		2533 ms (1,1%)	766214
org.jdom.Verifier.**isXMLLetter** (char)		2336 ms (1%)	5063797
metalism.core.agent.gui.AgentSupervisorGUI.**initInternalFrames** ()		2173 ms (0,9%)	113
metalism.core.agent.gui.Usher.**componentAdded** (java.awt.event.ContainerEvent)		1956 ms (0,8%)	51374
org.jdom.Verifier.**checkCharacterData** (String)		1684 ms (0,7%)	575219
org.jdom.AttributeList.**indexOf** (String, org.jdom.Namespace)		1452 ms (0,6%)	1274971
metalism.core.util.CognitionControl.**P** (boolean, int, int, int[], int[], Double[], Double[], metalism.core.util.UnificationMatrix)		1334 ms (0,6%)	1016
org.jdom.Attribute.**<init>** (String, String, int, org.jdom.Namespace)		1179 ms (0,5%)	574943
org.jdom.Verifier.**isHighSurrogate** (char)		1083 ms (0,5%)	2689555
metalism.core.util.RelationsGraph.**init** ()		1049 ms (0,4%)	8
org.jgraph.graph.DefaultCellViewFactory.**createVertexView** (Object)		1038 ms (0,4%)	40
org.jdom.AttributeList.**add** (int, org.jdom.Attribute)		972 ms (0,4%)	574943
metalism.core.util.UnificationValue.**n** (int, int)		866 ms (0,4%)	138752
metalism.comparison.engine.EngineMainWindow.**initComponents** ()		819 ms (0,3%)	1
metalism.core.util.CognitionControl.**finalP** (boolean, Double, Double, metalism.core.util.UnificationValue)		785 ms (0,3%)	65024
org.jdom.output.Format$DefaultEscapeStrategy.**shouldEscape** (char)		783 ms (0,3%)	935792
org.jdom.Attribute.**getNamespaceURI** ()		780 ms (0,3%)	1907875
org.jdom.input.SAXHandler.**startElement** (String, String, String, org.xml.sax.Attributes)		773 ms (0,3%)	89107
org.jdom.Verifier.**isXMLNameStartCharacter** (char)		770 ms (0,3%)	766214
metalism.core.util.UnificationValue.**totalGlobal** (int)		760 ms (0,3%)	65024
org.jdom.AttributeList.**add** (Object)		738 ms (0,3%)	574943
metalism.core.agent.gui.Usher.**descend** (boolean[][], java.awt.Dimension, java.awt.Component[], java.awt.Component, java.awt.Point)		719 ms (0,3%)	410306
org.jdom.Element.**<init>** (String, org.jdom.Namespace)		709 ms (0,3%)	191271
org.jdom.Verifier.**isXMLCharacter** (int)		703 ms (0,3%)	1753763
org.jdom.output.XMLOutputter.**printAttributes** (java.io.Writer, java.util.List, org.jdom.Element, org.jdom.output.XMLOutputter.NamespaceStack)		694 ms (0,3%)	102164
org.jdom.input.SAXBuilder.**build** (org.xml.sax.InputSource)		687 ms (0,3%)	2
metalism.comparison.engine.Boot.**writeUnificationRegistryXMLFile** (String, metalism.core.util.UnificationRegistry)		634 ms (0,3%)	2
jade.lang.acl.ACLMessage.**getContentObject** ()		618 ms (0,3%)	264
org.jdom.Element.**setAttribute** (String, String)		610 ms (0,3%)	307248

Рисунок 4.3.2 – Профілювання усього проекту

Ефективність роботи агентів представлена на малюнку 4.3.3, де реєструється загальний час виконання кожного агента. Ми можемо спостерігати, що кожному агенту для виконання свого завдання потрібен приблизний час 2100 с. Процес Observer вимагає майже 1200 с.



Call Tree - Method	Time [%]	Time	Invocations
AWT-EventQueue-0		41772 ms (100%)	1
MMAGENT[AGENT-emotional]		22910 ms (100%)	1
MMAGENT[ENVIR-Virtual_3D]		22805 ms (100%)	1
MMAGENT[AGENT-cognitive]		22732 ms (100%)	1
MMAGENT[INTER-Rules_based]		22701 ms (100%)	1
MMAGENT[ENVIR-Competence]		22528 ms (100%)	1
MMAGENT[ORGAN-Group_organization]		22376 ms (100%)	1
MMAGENT[INTER-Dialog_based]		22149 ms (100%)	1
MMAGENT[ORGAN-Hierarchical]		21909 ms (100%)	1
OBSERVER		13449 ms (100%)	1
Thread-10		1859 ms (100%)	1
Thread-9		66.5 ms (100%)	1
Thread-8		58.5 ms (100%)	1
main		32.1 ms (100%)	1

Рисунок 4.3.3 – Продуктивність профілювальних агентів

Порівнюючи наші результати з аналогічним дослідженням, де порівняння семи різних платформ досягається за допомогою аналогічної ситуації з аукціоном, ми можемо побачити, що тип агента, який використовується на кожній платформі, має риси агента BDI. Цей тип агента пов'язаний з типом когнітивного агента, обраного в нашому підході. Однак у нашому кейсі використовується обмежений

набір функцій, і не всі вони є на всіх платформах і навпаки. Проте ми бачимо, що наші рішення виявилися найближчими до найбільш підходящих. Таким чином, це свідчить про те, що нинішній підхід, заснований на самоорганізованій БПС, є перспективним, і ми продовжуватимемо працювати над удосконаленням та вдосконаленням цього підходу.

4.4 Висновок четвертого розділу

У четвертому розділі магістерської роботи виконано практичну реалізацію прототипу інструмента підтримки прийняття рішень для проєктування багатоагентних програмних систем, а також проведено його тестування й валідацію на прикладному кейсі. Розділ логічно поєднує три взаємопов’язані компоненти: реалізацію графічного інтерфейсу та програмних модулів, експериментальну перевірку коректності процесів AS-характеризації та MM-характеристики, а також аналіз результатів і продуктивності системи на основі профілювання. Таким чином, підтверджено прикладну життєздатність запропонованого підходу та його придатність до використання як інженерного інструмента.

У частині реалізації програмного застосунку ключовим результатом стало створення зручного GUI, який структуровано відповідно до логіки методу та відображає основні кроки процесу узгодження. Інтерфейс організовано у вигляді двох головних модулів: генератора бази знань (із підмодулями редактора доменів/проблемних характеристик та менеджера мета-моделей) і модуля узгодження, що забезпечує повний цикл зіставлення. Важливо, що реалізація передбачає не лише виконання алгоритмів, а й прозору взаємодію користувача з даними: завантаження КВ, вибір або визначення домену, формування/завантаження мікромасиву, відбір кандидатних мета-моделей та запуск механізму узгодження з подальшим відображенням проміжних і фінальних результатів. Додатковою практично значущою деталлю є використання значення 0,5 за замовчуванням для нових зв’язків «характеристика–домен», що інтерпретується як максимальна

невизначеність і створює коректну стартову позицію для подальшого уточнення знань.

Суттєвою перевагою реалізованого прототипу є наявність засобів спостереження за перебігом колективного прийняття рішень у БПС. Модуль перегляду (MAS-Observer) дозволяє аналізувати внутрішній стан агентів під час формування груп рішень, що підсилює інтерпретованість результатів, спрощує налагодження та робить систему корисною не лише як «чорна скринька», а як інструмент із пояснювальними можливостями. Також реалізовано відображення результатів у вигляді діаграм із двома ключовими вимірами – впевненості щодо відповідності специфікації застосунку та впевненості сумісності групи – що забезпечує наочну оцінку компромісів і якості вибору.

У частині тестування та валідації продемонстровано роботу системи на кейсі, що моделює віртуальну торгову платформу з механізмами аукціону, подібну за класом задач до Amazon-подібних сценаріїв. Валідація виконана послідовно за етапами методу: спершу – приклад AS-характеризації, де на основі виявлених характеристик і значень належності обґрунтовано вибір домену рішення. Показано, що з використанням статистичних показників (середнє, медіана, дисперсія та стандартне відхилення) система зменшує невизначеність і локалізує домен із найбільшою правдоподібністю (у наведеному прикладі – $\delta 3$). Отже, підтверджено, що механізм характеризації формує якісний вхід для наступних етапів і забезпечує відтворювану логіку вибору навіть за скороченого набору доменів і ознак.

Далі виконано валідацію ММ-характеристики та наповнення КБ мета-моделей через абстрагування ознак і задання значень ефективності на основі аналізу джерел та експертних даних. Продемонстровано, що використання ефективностей у розрізі доменів дозволяє перейти від описового рівня (перелік функцій Ф) до кількісного рівня придатності мета-моделей у конкретній проблемній області. Додатково в розділі показано застосування емпіричного досвіду у вигляді гістограм (за результатами опитувань) та баєсового висновування для оцінювання сумісності, що забезпечує формалізовану основу для рішень агентів під час переговорів і формування груп.

У результаті експерименту підтверджено працездатність колективного механізму відбору: агенти формують групи, оцінюючи ймовірність сумісності можливих партнерів і сумісність групи з проблемою, після чого виконують колективний вибір найкращої групи. Для наведеного кейсу найбільш сумісною виявилась комбінація мета-моделей: когнітивний агент, компетентнісне середовище, діалогова взаємодія та ієрархічна організація, що узгоджується з логікою аукціонних і переговорних сценаріїв та підтверджує адекватність отриманих рішень.

Окремий практичний підсумок розділу – аналіз продуктивності за допомогою профайлера NetBeans. Виявлено, що найбільш ресурсоємними операціями є читання/запис до реєстру уніфікації, який інтенсивно використовується агентами для перевірки сумісності; наступними за значущістю є процедури «value» та «compare», безпосередньо пов’язані з оцінюванням відповідностей. Також оцінено часові витрати на виконання агентів та процесу спостерігача, що дозволяє визначити «вузькі місця» і сформувати напрямки оптимізації (зокрема – оптимізація доступу до реєстру уніфікації та зменшення частоти звернень до спільних структур).

Загалом, у розділі підтверджено, що реалізований прототип є функціонально завершеним для демонстрації запропонованого підходу: він підтримує створення й редагування баз знань, виконує AS- та MM-характеристики, забезпечує відбір мета-моделей через багатоагентну взаємодію та надає засоби візуалізації, валідації й аналізу продуктивності. Отримані результати свідчать про перспективність самоорганізованого багатоагентного підходу для задач узгодження та добору компонентів БПС, а також формують основу для подальшого розвитку системи — розширення КВ, автоматизації наповнення знань та оптимізації продуктивності механізму узгодження.

ВИСНОВКИ

У магістерській роботі виконано комплексне опрацювання предметної області багатокомпонентних програмних систем і обґрунтовано доцільність застосування агентно-орієнтованої архітектури для зниження складності проєктування та підтримки прийняття рішень на ранніх етапах розробки. У ході дослідження проаналізовано сучасні наукові підходи та сформовано вимоги до методу і програмного інструмента, що дозволяє системно узгоджувати специфікацію задачі із набором можливих мета-моделей та компонентів рішення.

Ключовим результатом роботи є розроблення методу та механізму проєктування ПЗ на основі агентно-орієнтованої архітектури для багатокомпонентних систем, у межах якого реалізовано ідею «суспільства агентів-приймачів рішень»: кожен агент оцінює окремий аспект моделі та бере участь у колективному формуванні узгодженого набору компонентів. Уперше в межах цього підходу поєднано характеристики агентів, специфікації задачі та використання мета-моделей, що забезпечує більш обґрунтований вибір поведінки агентів і варіанта архітектурної інтеграції в загальній структурі системи.

Для реалізації запропонованого підходу систематизовано логіку механізму добору мета-моделей: від формування AS-мікромасиву та визначення домену до відбору кандидатних MM-MicroArray, створення агентів-кандидатів, переговорного формування груп рішень, оцінювання сумісності та голосування за найкращу групу. Також формалізовано MM-характеристику як процес вилучення ознак мета-моделі та приєднання статистичних значень ефективності в доменах, що робить можливим кількісне порівняння альтернатив і раціональний вибір компонентів рішення.

Практичну придатність методу підтверджено створенням прототипу програмного застосунку, який реалізує повний сценарій «характеризація → відбір → узгодження → візуалізація результату». Прототип підтримує створення й редагування баз знань, виконує AS- та MM-характеристики, забезпечує відбір мета-моделей через багатоагентну взаємодію та надає засоби валідації й аналізу продуктивності.

У межах тестування та валідації проведено кейс-стаді, що продемонструвало здатність системи зменшувати початкову невизначеність: AS-характеристика дозволяє локалізувати домен задачі, після чого механізм узгодження формує узгоджену групу мета-моделей на основі оцінок сумісності, переговорів і колективного рішення агентів. Отримані результати підтверджують перспективність самоорганізованого багатоагентного підходу для задач узгодження та добору компонентів БПС.

Окремим практичним результатом є оцінювання продуктивності (на основі профілювання), що дозволило виявити «вузькі місця» та визначити напрямки оптимізації: найбільш ресурсоємними є операції читання/запису до реєстру уніфікації та процедури оцінювання відповідностей (типу value/compare), які інтенсивно використовуються агентами під час перевірки сумісності та формування груп рішень.

Практичне значення роботи полягає в тому, що запропонований метод і механізм узгодження зменшують ризик помилок вибору методології/моделі, підтримують прийняття рішень на стартових кроках розробки й можуть виступати інструментом підтримки проєктувальника під час побудови багатокомпонентних агентних систем.

Перспективи подальших досліджень пов’язані з: (1) розширенням і автоматизацією наповнення баз знань (зокрема, за рахунок аналізу документів/веб-джерел), (2) підтримкою динамічної композиції веб-сервісів і додаванням нових агентів без перебудови системи, (3) стандартизацією подання специфікацій та результатів характеризації (XML-орієнтовані формати на кшталт ADMACA), (4) оптимізацією продуктивності механізму узгодження.

Як підсумок, робота демонструє працездатний і розширюваний підхід до формалізованого добору мета-моделей та компонентів рішення на основі багатоагентної взаємодії, що підвищує керованість процесу проєктування та знижує початкову невизначеність у задачах побудови багатокомпонентних програмних систем.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Grassi S, A., Guizzi, G., Santillo, L. C., & Vespoli, S. A semi-heterarchical production control architecture for industry 4.0-based manufacturing systems. Manufacturing Letters, 24. 2020. p.43–46. URL: https://doi.org/10.1016/j.mfglet.2020.03.007.

2. Grassi S, A.; Guizzi, G.; Santillo, L.C.; Vespoli, S. The manufacturing planning and control system: A journey towards the new perspectives in industry 4.0 architectures. Int. Ser. Oper. Res. Manag. Sci. 2020, p.193–216.

3. Nele, L.; Mattera, G.; Yap, E.W.; Vozza, M.; Vespoli, S. Towards the application of machine learning in digital twin technology: A multi-scale review. Discov. Appl. Sci. 2024. p. 502.

4. Panzer, M.; Bender, B. Deep reinforcement learning in production systems: A systematic literature review. Int. J. Prod. Res. 2021, 60, p. 316–341.

5. Khadivi, M.; Charter, T.; Yaghoubi, M.; Jalayer, M.; Ahang, M.; Shojaeinasab, A.; Najjaran, H. Deep reinforcement learning for machine scheduling: Methodology, the state-of-the-art, and future directions. Comput. Ind. Eng. 2025, p. 200,

6. Revetria, R.; Damiani, L.; Rozhok, A.; Ahmad, K. Use of Modeling & Simulation and AI for Collaborative Framework: SMEs Supply Chain Disruptions. Front. Artif. Intell. Appl. 2024, p.392–397

7. Vespoli, S.; Grassi, A.; Guizzi, G.; Popolo, V. Generalised Performance Estimation in Novel Hybrid MPC Architectures: Modeling the CONWIP Flow-Shop System. Appl. Sci. 2023, 13, p.808.

8. Vespoli, S.; Grassi, A.; Guizzi, G.; Popolo, V. A Deep Learning Algorithm for the Throughput Estimation of a CONWIP Line. In Proceedings of the IFIP Advances in Information and Communication Technology, Nantes, France, 5–9 September 2021; Volume 630, pp. 143–151.

9. Marchesano, M.G.; Guizzi, G.; Santillo, L.C.; Vespoli, S. A Deep Reinforcement Learning approach for the throughput control of a Flow-Shop production system. IFAC-PapersOnLine 2021. P. 61–66.

10. Vespoli, S.; Mattera, G.; Marchesano, M.G.; Nele, L.; Guizzi, G. Adaptive manufacturing control with Deep Reinforcement Learning for dynamic WIP management in industry 4.0. Comput. Ind. Eng. 2025.p.966

11. Marchesano, M.G.; Guizzi, G.; Santillo, L.C.; Vespoli, S. Dynamic Scheduling in a Flow Shop Using Deep Reinforcement Learning. In Proceedings of the IFIP Advances in Information and Communication Technology, Nantes, France, 5–9 September 2021; Volume 630, pp. 152–160.

12. Ahmad, K.; Rozhok, A.; Revetria, R. Supply Chain Resilience in SMEs: Integration of Generative AI in Decision-Making Framework. In Proceedings of the 2024 International Conference on Machine Intelligence and Smart Innovation (ICMISI 2024), Alexandria, Egypt. 12–14 May 2024. pp. 295–299.

13. Zhang, C.; Juraschek, M.; Herrmann, C. Deep reinforcement learning-based dynamic scheduling for resilient and sustainable manufacturing: A systematic review. J. Manuf. Syst. 2024. pp. 962–989.

14. Spearman, M.L.; Woodruff, D.L.; Hopp, W.J. CONWIP Redux: Reflections on 30 years of development and implementation. Int. J. Prod. Res. 2022. p.381–387.

15. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. Nature 2020.p.529–533.

16. Park, I.B.; Huh, J.; Kim, J.; Park, J. A Reinforcement Learning Approach to Robust Scheduling of Semiconductor Manufacturing Facilities. IEEE Trans. Autom. Sci. Eng. 2020. p.420–431.

17. Wang, H.; Sarker, B.R.; Li, J.; Li, J. Adaptive scheduling for assembly job shop with uncertain assembly times based on dual Q-learning. Int. J. Prod. Res. 2020. p.867–883.

18. Li, F.; Lang, S.; Hong, B.; Reggelin, T. A two-stage RNN-based deep reinforcement learning approach for solving the parallel machine scheduling problem with due dates and family setups. J. Intell. Manuf. 2023. p. 107–140.

19. Thürer, M.; Stevenson, M.; Silva, C.; Land, M.J.; Fredendall, L.D. Workload Control and Order Release: A Lean Solution for Make-to-Order Companies. Prod. Oper. Manag. 2012. p.939–953.

20. Xia, K.; Sacco, C.; Kirkpatrick, M.; Saidy, C.; Nguyen, L.; Kircaliali, A.; Harik, R. A digital twin to train deep reinforcement learning agent for smart manufacturing plants: Environment, interfaces and intelligence. J. Manuf. Syst. 2020. p.210–230.

21. ang, W.; Zhang, Y.; Wang, Y.; Pan, G.; Feng, Y. Hierarchical multi-agent deep reinforcement learning for dynamic flexible job-shop scheduling with transportation. Int. J. Prod. Res. 2025, p.1–28.

22. Kaven, L.; Huke, P.; Göppert, A.; Schmitt, R.H. Multi agent reinforcement learning for online layout planning and scheduling in flexible assembly systems. J. Intell. Manuf. 2024, 35, 3917–3936.

23. Panzer, M.; Gronau, N. Designing an adaptive and deep learning based control framework for modular production systems. J. Intell. Manuf. 2023. p. 113–136.

24. Dittrich, M.A.; Fohlmeister, S. Cooperative multi-agent system for production control using reinforcement learning. CIRP Ann. 2020. p.389–392.

25. Oukil, A.; El-Bouri, A. Ranking dispatching rules in multi-objective dynamic flow shop scheduling: A multi-faceted perspective. Int. J. Prod. Res. 2021. p.388–411.

26. Park, J.; Chun, J.; Kim, S.H.; Kim, Y.; Park, J. Learning to schedule job-shop problems: Representation and policy learning using graph neural network and reinforcement learning. Int. J. Prod. Res. 2021.p. 360–377.

27. Vespoli, S.; Guizzi, G.; Gebennini, E.; Grassi, A. A novel throughput control algorithm for semi-heterarchical industry 4.0 architecture. Ann. Oper. Res. 2021. p.201–221

28. Wenzhi, S.; Zhang, H.; Tseng, M.-L.; Weipeng, Z.; Xinyang, L. Hierarchical Energy Optimization Management of Active Distribution Network with Multi-Microgrid System. J. Ind. Prod. Eng. 2022. p. 210–229.

29. Iftikhar, M.Z.; Imran, K. Network Reconfiguration and Integration of Distributed Energy Resources in Distribution Network by Novel Optimization Techniques. Energy Rep. 2024. p.155–179.

30. Xu, X.; Zhao, N.; Wang, L.; Yao, X.; Zhou, L. Research on Time-Varying Dynamic Response Aggregation Model of Distributed Generator Participating in Active Distribution Network. Energy Rep. 2023. p. 546–556.

31. Chen, Y.; Liu, Y.; Shen, X.; Xu, L.; Liu, J.; Li, L. Review of Edge Intelligence Technology for Distributed Energy Resources in Urban Energy Systems. Autom. Electr. Power Syst. 2022.p.142–152.

32. Shen, S.; Ren, Y.; Ju, Y.; Wang, X.; Wang, W.; Leung, V.C.M. EdgeMatrix: A Resource-Redefined Scheduling Framework for SLA-Guaranteed Multi-Tier Edge-Cloud Computing Systems. IEEE J. Sel. Areas Commun. 2023. p. 820–834.

33. Liu, Z.; Song, J.; Qiu, C.; Wang, X.; Chen, X.; He, Q.; Sheng, H. Hastening Stream Offloading of Inference via Multi-Exit DNNs in Mobile Edge Computing. IEEE Trans. Mob. Comput. 2024. p.535–548.

34. Nie, Y.; Peng, C.; Hu, Y.; He, Y.; Ma, G.; Huang, C. Parallel distributed optimal economic dispatch of High penetration microgrid based on edge computing. South. Power Syst. Technol. 2023. p.114–124.

35. Jie, T.; Qi, Z.; Pu, T.; Song, R.; Zhang, J.; Tan, Y.; Wang, X. Edge Intelligence in Power Internet of Things: Concept, Architecture, Technology and Application. Proc. CSEE 2024. 2024. p.473–496.

36. hang, D.; Li, X.; Tao, W. Location of single-phase grounding fault segment in limited information distribution network based on edge computing and deep learning. Power Syst. Prot. Control 2023. p. 22–32.

37. Qin, Q.; Liu, W.; Tan, W.; Cai, Z.; Cen, B.; Kuang, P. Optimal deployment method of distribution edge computing terminal for software defined network. Electr. Power Constr. 2023 p. 82–90.

38. an, S.; Liu, B. Optimal allocation method of edge computing unit for fast Fault processing of distribution network. Electr. Power Constr. 2022.p.31–41.

39. Liu, D.; Zeng, X.; Wang, Y. Control Strategy of virtual power station in Distribution area based on Edge Computing Architecture. Trans. China Electrotech. Soc. 2021. p. 2852–2860.

40. Yuan, L.; Gu, J.; Jin, Z. User-side data application framework of Power Internet of Things based on Cloud-edge-end collaboration. Electr. Power Constr. 2020. p.1–8.

ДОДАТКИ

ДОДАТОК А

(обов’язковий)

КОПІЯ НАУКОВОЇ ПУБЛІКАЦІЇ

Міністерство освіти і науки України
Хмельницький національний університет

ЗБІРНИК НАУКОВИХ ПРАЦЬ

за матеріалами XVII Всеукраїнської науково-практичної конференції
«Актуальні проблеми комп'ютерних наук АПКН-2025»

14-15 листопада 2025

Хмельницький 2025

УДК 004.41

Горбатюк І.В., Форкун Ю.В.

Хмельницький національний університет

МЕТОД ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ АГЕНТНО-ОРІЄНТОВАНОЇ АРХІТЕКТУРИ ДЛЯ БАГАТОКОМПОНЕНТНИХ ПРОГРАМНИХ СИСТЕМ

У роботі розглянуто агентно-орієнтований метод проєктування програмного забезпечення для багатокомпонентних систем. Розроблено процес автоматичного вибору метамоделей і конфігурацій агентів із використанням баєсового пізнання та мультиагентного підходу. Підхід забезпечує узгодженість компонентів, скорочує час проєктування та підвищує ефективність розробки складних програмних систем.

The paper considers an agent-oriented method for software design for multi-component systems. A process for automatic selection of metamodels and agent configurations using Bayesian learning and a multi-agent approach is developed. The approach ensures component consistency, reduces design time, and increases the efficiency of developing complex software systems.

Сучасні інформаційні системи стають дедалі складнішими, поєднуючи різні програмні компоненти, що взаємодіють між собою в умовах децентралізованого управління. Такі системи вимагають нових методів проєктування, орієнтованих на автономність, адаптивність і координацію між частинами. Одним із ефективних підходів є агентно-орієнтована архітектура, яка забезпечує гнучкість, масштабованість і підвищену надійність при створенні багатокомпонентних систем.

Робота присвячена розробці методу проєктування програмного забезпечення, заснованого на агентно-орієнтованій парадигмі. Основна ідея полягає у використанні мультиагентних систем (МАС) для моделювання взаємодії між компонентами програмного комплексу. Кожен агент виступає як автономна сутність із власними цілями, знаннями та можливістю співпраці з іншими агентами.

Метою дослідження є створення автоматизованого механізму вибору оптимальної конфігурації агентів та метамоделей для побудови рішення, що відповідає заданій проблемі. Такий механізм аналізує опис проблеми, зіставляє її характеристики з базою знань про існуючі агентні метамоделі й обирає найбільш сумісний набір компонентів для побудови системи.

У межах дослідження розроблено процес, який складається з трьох основних етапів:

– характеризація опису проблеми та визначення домену рішення;

– оцінювання продуктивності наявних метамоделей для цього домену;

– автоматичне зіставлення проблеми та набору метамоделей за допомогою мультиагентного підходу.

Запропонована архітектура мультиагентної системи містить агенти, які представляють окремі метамоделі та взаємодіють між собою, формуючи групу найбільш сумісних моделей. Кожен агент використовує критерії сумісності для узгодження рішень з іншими агентами. У результаті формується узгоджена група метамоделей, що становить основу запропонованого рішення.

Розроблено прототип інструменту, що реалізує запропонований метод і дозволяє автоматизувати вибір архітектурних компонентів. Для цього застосовано методи кооперативного штучного інтелекту, баєсового пізнання та метааналізу, що враховують невизначеність і статистичні закономірності у прийнятті рішень.

Практичне значення дослідження полягає у можливості використання запропонованого методу для створення складних систем — від сенсорних мереж до автономних керуючих комплексів. Використання агентного підходу дозволяє підвищити надійність програмних систем, забезпечити їх самодостатність і спроможність до адаптації в умовах динамічних середовищ.

Проведені експерименти показали, що застосування розробленого механізму скорочує час проєктування в середньому на від 5 до 15%, зменшує кількість помилок узгодження між компонентами та підвищує стабільність системи під час розгортання.

Запропонований підхід може бути використаний у процесах інженерії програмного забезпечення, особливо при створенні розподілених, адаптивних і високонадійних програмних рішень. Його гнучкість дозволяє масштабувати архітектуру та повторно використовувати окремі компоненти в інших проєктах.

Подальші дослідження планується спрямувати на вдосконалення інструменту автоматичного аналізу, розширення бази знань про метамоделі, а також інтеграцію з сучасними технологіями машинного навчання для підвищення точності рішень і прогнозування ефективності компонентів.

Перелік посилань

1. Blum J. Exploring Arduino: Tools and Techniques for Engineering Wizardry. 2nd ed. Wiley. 2021.
2. McGarry, S. What is Schematic Design? Understanding Schematics and Their Importance. URL: https://www.autodesk.com/products/fusion-360/blog/schematic-design/
3. Law A. M. Simulation Modeling and Analysis. 6th ed. New York : McGraw-Hill Education, 2023. 864 p.
4. Бабенко В. О., Попова О. В. Застосування методів системного аналізу та експертних оцінок для оптимізації управлінських рішень. Економіка та суспільство. 2023. № 49.
5. Матвійків О., Ткаченко С., Хаханов В. Інженерне проєктування складних об'єктів і систем. – Львів: НУ «Львівська політехніка», 2021. – 260 с.
6. Lemmon M.D. Lectures for Advanced Control Systems. University of Notre Dame, 2025. 346 p. URL: https://www3.nd.edu/~lemmon/courses/advanced-control/lecture-book/advanced-control-2025.pdf.

ДОДАТОК Б

(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Метод проєктування програмного забезпечення на основі агентно-орієнтованої архітектури для багатокомпонентних програмних систем

Виконав : студент групи ІПЗм-24-1
Горбатюк Іван Володимирович

Керівник : канд. техн. наук, доцент
Форкун Юрій Вікторович

Рисунок Б.1 – Слайд 1

АКТУАЛЬНІСТЬ ТЕМИ

2

- Сучасні програмні системи стають багатокомпонентними, розподіленими та складними.
- Традиційні підходи до проєктування не завжди забезпечують гнучкість і масштабованість.
- Агентно-орієнтована архітектура дозволяє будувати автономні та адаптивні компоненти.
- Вибір методології та мета-моделі проєктування є складним на початкових етапах.
- Потрібні інструменти підтримки прийняття рішень під час проєктування.

Рисунок Б.2 – Слайд 2

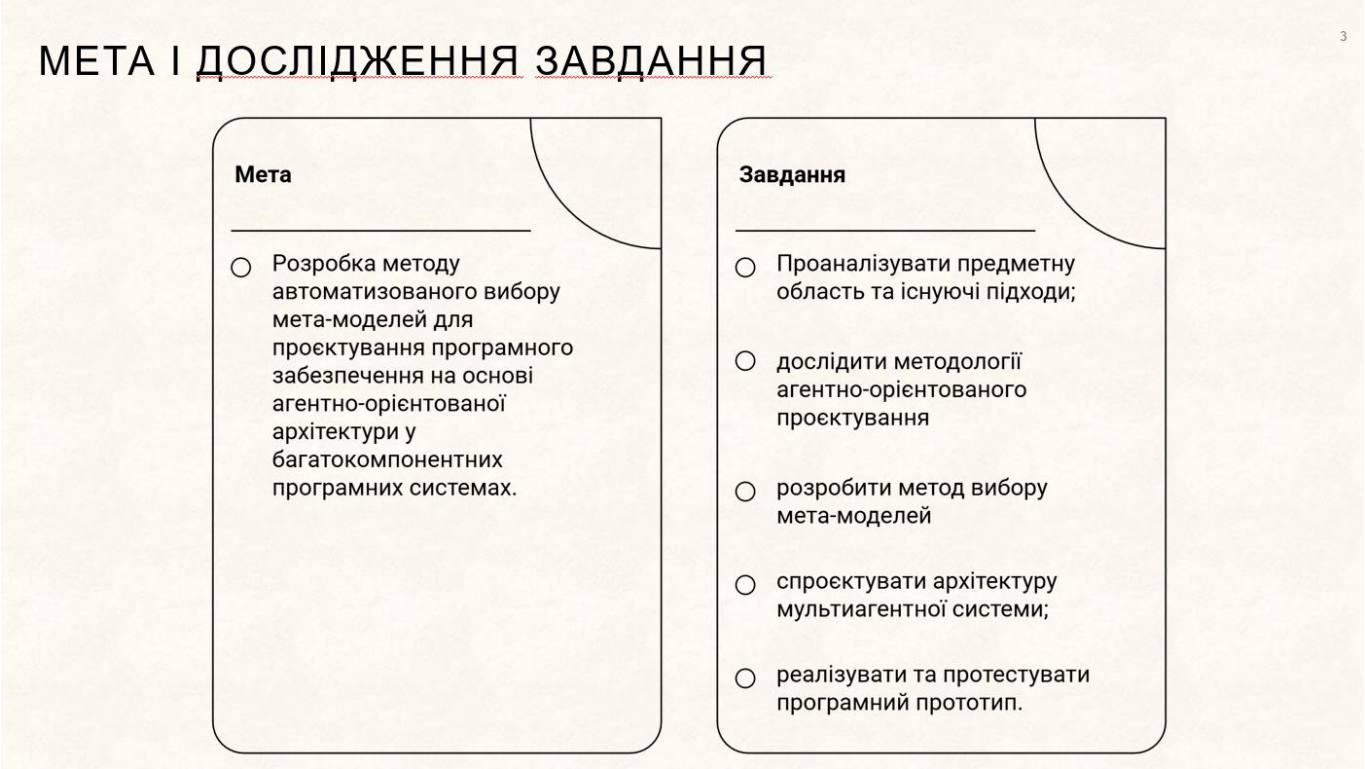


Рисунок Б.3 – Слайд 3

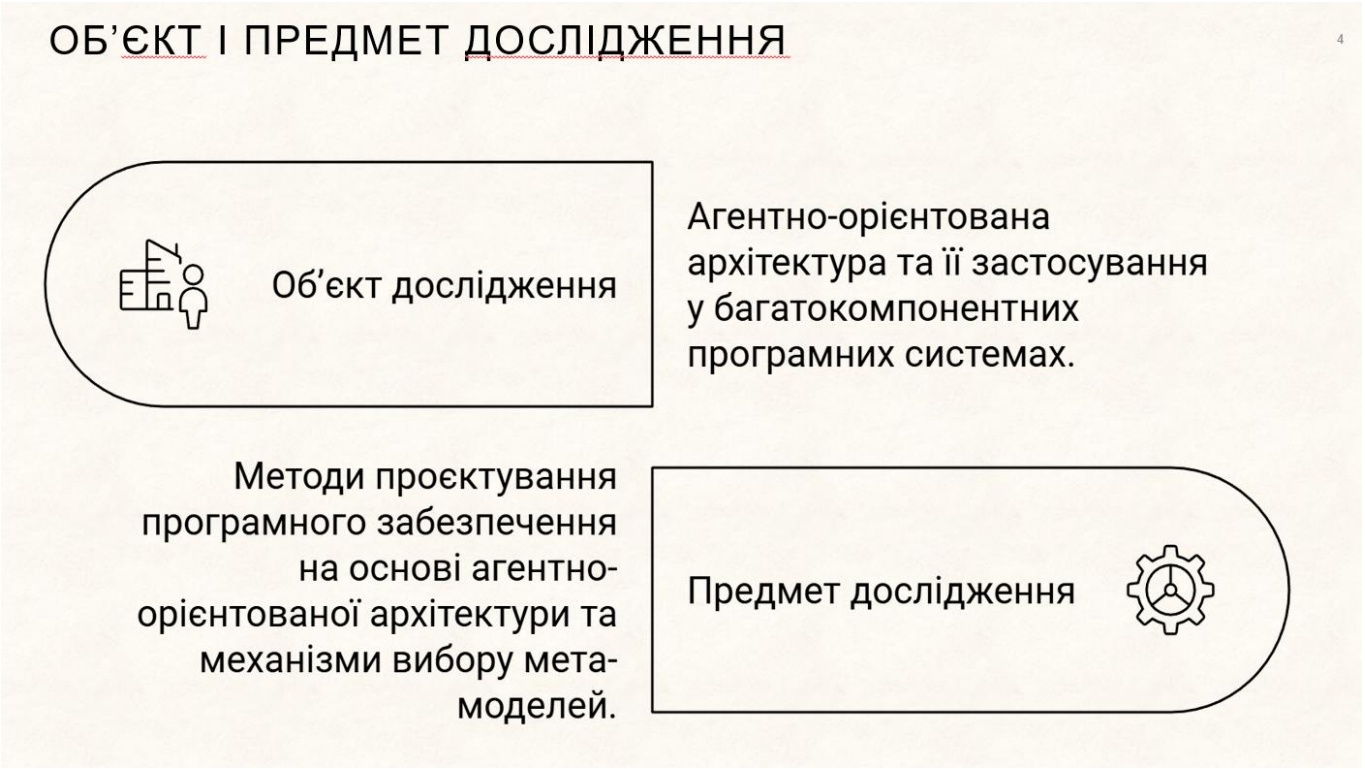


Рисунок Б.4 – Слайд 4

МЕТОДИ ДОСЛІДЖЕННЯ

5

- Аналіз і синтез — для дослідження предметної області та формування вимог.
- Порівняльний аналіз — для оцінювання існуючих методологій агентно-орієнтованого проєктування.
- Моделювання — для побудови архітектури мультиагентної системи та мета-моделей.

Рисунок Б.5 – Слайд 5

НАУКОВА НОВИЗНА

6

- Запропоновано метод проєктування програмного забезпечення на основі агентно-орієнтованої архітектури для багатокомпонентних програмних систем.
- Розроблено підхід із використанням агентів-приймачів рішень для вибору мета-моделей.
- Кожен агент здійснює оцінювання окремих характеристик системи та вимог застосунку.
- Забезпечено обґрунтований та узгоджений вибір архітектурних рішень.

Рисунок Б.6 – Слайд 6

РОЗДІЛ 1 — ХАРАКТЕРИСТИКА ПРЕДМЕТНОЇ ОБЛАСТІ

7

- Багатокомпонентні програмні системи складаються з автономних програмних компонентів, що взаємодіють між собою.
- Такі системи характеризуються децентралізованою структурою та відсутністю єдиного центру керування.
- Агентно-орієнтований підхід дозволяє представляти компоненти системи у вигляді агентів з власною поведінкою.
- Агенти можуть самостійно приймати рішення та взаємодіяти для досягнення спільної мети.

Рисунок Б.7 – Слайд 7

РОЗДІЛ 1 — ПРОБЛЕМИ ПРОЄКТУВАННЯ БАГАТОКОМПОНЕНТНИХ ПРОГРАМНИХ СИСТЕМ

8

- Зростання кількості компонентів ускладнює процес проєктування системи.
- Важко передбачити emergent-поведінку агентів у процесі їх взаємодії.
- Вибір методології агентно-орієнтованого проєктування залежить від предметної області та вимог системи.
- Відсутній універсальний підхід до вибору мета-моделей проєктування.

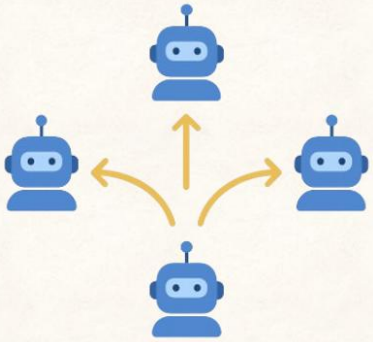


Рисунок Б.8 – Слайд 8

Рисунок Б.9 – Слайд 9

Рисунок Б.10 – Слайд 10

Рисунок Б.11 – Слайд 11

Рисунок Б.12 – Слайд 12

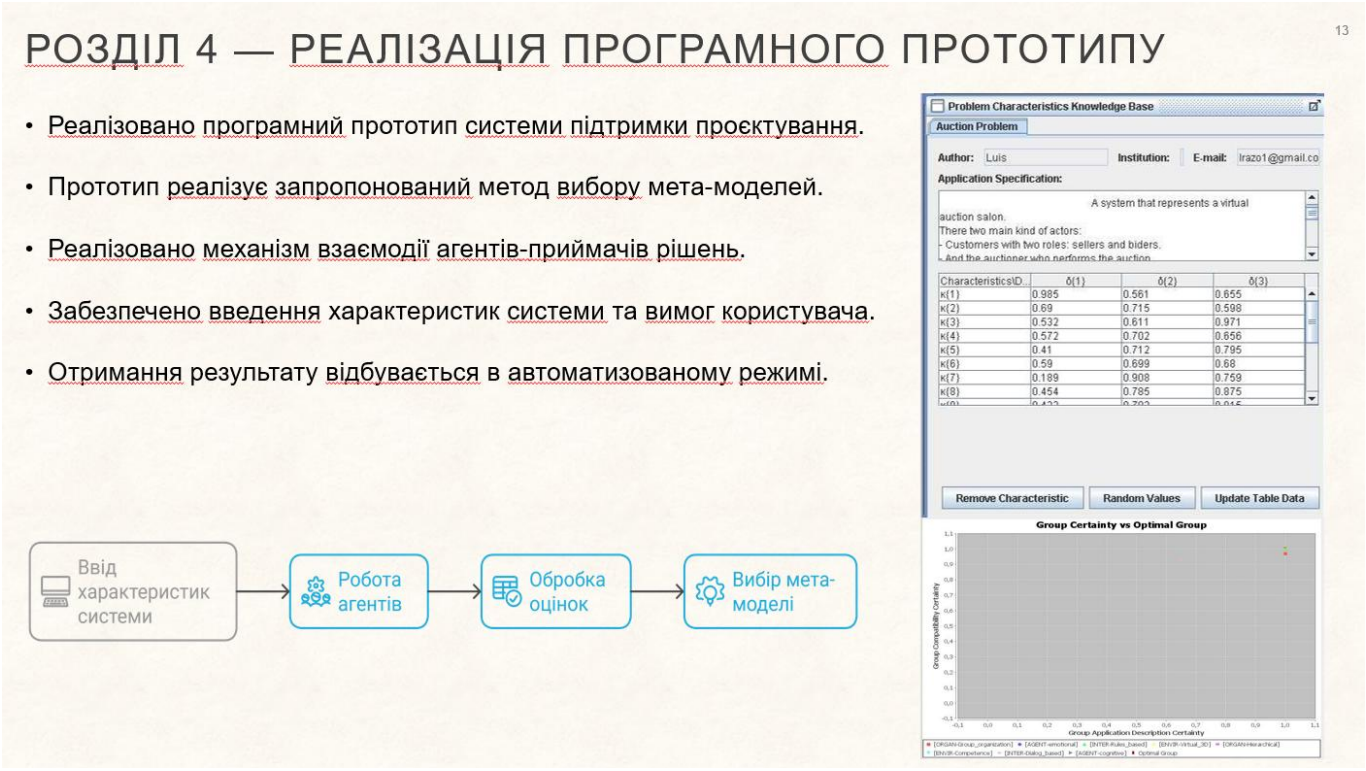


Рисунок Б.13 – Слайд 13

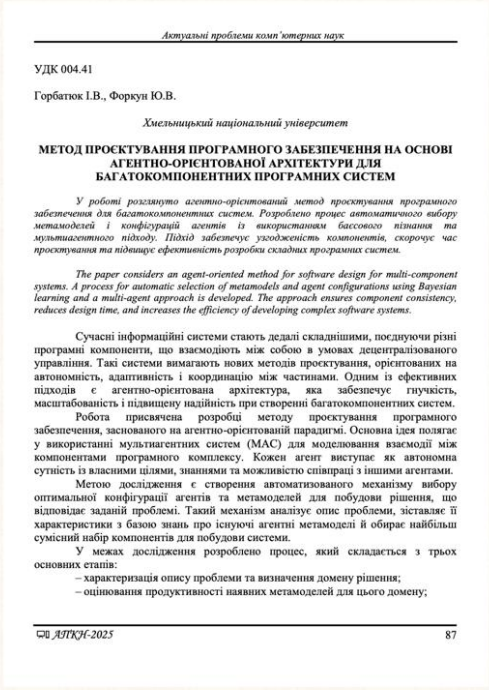
Рисунок Б.14 – Слайд 14

ПУБЛІКАЦІЇ

15

- Горбатюк І.В., Форкун Ю.В.
Метод проєктування програмного забезпечення на основі агентно-орієнтованої архітектури для багатокомпонентних програмних систем

Актуальні проблеми комп'ютерних наук (АПКН–2025), Хмельницький, 2025 (87).



Актуальні проблеми комп'ютерних наук

УДК 004.41

Горбатюк І.В., Форкун Ю.В.

Хмельницький національний університет

МЕТОД ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ АГЕНТНО-ОРІЄНТОВАНОЇ АРХІТЕКТУРИ ДЛЯ БАГАТОКОМПОНЕНТНИХ ПРОГРАМНИХ СИСТЕМ

У роботі розглянуто агентно-орієнтований метод проєктування програмного забезпечення для багатокомпонентних систем. Розроблено процес автоматичного вибору метамоделей і конфігурацій агентів із використанням баєсового навчання та мультиагентного підходу. Підхід забезпечує узгодженість компонентів, скорочує час проєктування та підвищує ефективність розробки складних програмних систем.

The paper considers an agent-oriented method for software design for multi-component systems. A process for automatic selection of metamodels and agent configurations using Bayesian learning and a multi-agent approach is developed. The approach ensures component consistency, reduces design time, and increases the efficiency of developing complex software systems.

Сучасні інформаційні системи стають дедалі складнішими, поєднуючи різні програмні компоненти, що взаємодіють між собою в умовах децентралізованого управління. Такі системи вимагають нових методів проєктування, орієнтованих на автономність, адаптивність і координацію між частинами. Одним із ефективних підходів є агентно-орієнтована архітектура, яка забезпечує гнучкість, масштабованість і підвищену надійність при створенні багатокомпонентних систем.

Робота присвячена розробці методу проєктування програмного забезпечення, заснованого на агентно-орієнтованій парадигмі. Основна ідея полягає у використанні мультиагентних систем (МАС) для моделювання взаємодії між компонентами програмного комплексу. Кожен агент виступає як автономна сутність із власними цілями, знаннями та можливістю співпраці з іншими агентами.

Метою дослідження є створення автоматизованого механізму вибору оптимальної конфігурації агентів та метамоделей для побудови рішення, що відповідає заданій проблемі. Такий механізм аналізує опис проблеми, зіставляє її характеристики з базою знань про існуючі агентні метамоделі й обирає найбільш сумісний набір компонентів для побудови системи.

У межах дослідження розроблено процес, який складається з трьох основних етапів:

– характеризація опису проблеми та визначення домену рішення;
– оцінювання продуктивності наявних метамоделей для цього домену;

Рисунок Б.15 – Слайд 15

ВИСНОВКИ

Мета досягнута

Показано ефективність агентно-орієнтованого підходу для проєктування багатокомпонентних ПС.

Новизна

Вперше поєднано: характеристики агентів + специфікації задач + мета-моделі. Реалізовано «суспільство агентів-приймачів рішень» для колективного формування узгоджених компонентів.

Результат / Практичне застосування

Розроблено прототип ПЗ: характеристика → відбір → узгодження → візуалізація. Метод зменшує невизначеність, підтримує прийняття рішень та оптимізує добір компонентів системи.

Рисунок Б.16 – Слайд 16

Завідувачу кафедри
інженерії програмного забезпечення
проф. Бедратюку Л. П.
студента групи ІПЗм-24-1
Горбатюка І.В.

ЗАЯВА

Прошу закріпити за мною тему дипломної роботи освітнього ступеня «магістр» за спеціальністю 121 «Інженерія програмного забезпечення»: «Метод проєктування програмного забезпечення на основі агентно-орієнтованої архітектури для багатокомпонентних програмних систем»

(керівник дипломного проекту – к.т.н., доцент Форкун Юрій Вікторович)

30.06.25
Дата

Підпис студента

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти

Горбатюка І.В.

Прізвище, ініціали

факультет ІТ, 2 курс, група ІПЗм-24-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

30.06.25

дата

підпис

Mon Dec 15 05:06:23 EET 2025, Форкун Юрій Вікторович, Хмельницький національний університет, ХНУ

Anti-Plagiarism (UA) v-16.693

The maximum coincidence with one document 2.0%

Dictionaries check: UA, US, RU. **Errors in the documents: 7%**

	Document		Sum coincidence on the DB	
ID: 252808 Title: МКР_Метод проєктування програмного забезпечення на основі агентно-орієнтованої архітектури для багатокомпонентних програмних систем Added in a DB: 2025-12-15 Authors: Іван ГОРБАТЮК Heads: канд. техн. наук, доцент Юрій ФОРКУН Consultants: Opponents:	Symbols	Lexemes	Symbols	Lexemes
	131124	1808	5700 (4%)	54 (3%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Іван ГОРБАТЮК

Співавтор:

Назва: Метод проєктування програмного забезпечення на основі агентно-орієнтованої архітектури для багатокомпонентних програмних систем

Експерт: канд. техн. наук, доцент Юрій ФОРКУН

Підрозділ: Кафедра інженерії програмного забезпечення

Коефіцієнт подібності 1:1.7%

Коефіцієнт подібності 2:0.7%

Мікропробіли: 0

Заміна букв: 60

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-12-15 04:19:55.0

Після аналізу Звіту подібності констатую наступне:

☑ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

Дата 15.12.2025 експерт (Форкун Ю.В.)

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНИЙ ПРОЕКТ
освітнього ступеня «Магістр»

Дипломник Горбатюк Іван Володимирович

Тема Метод проєктування програмного забезпечення на основі агентно-орієнтованої архітектури для багатокомпонентних програмних систем

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг дипломного проекту:

Кількість листів креслень _____; кількість сторінок записки __102_____

1.Короткий зміст пояснювальної записки та прийнятих рішень У роботі досліджено агентно-орієнтовану архітектуру та методи проєктування багатокомпонентних програмних систем. Запропоновано новий метод проєктування агентів-приймачів рішень, що оцінюють окремі аспекти моделі та забезпечують автоматизований вибір мета-моделей. Розроблений підхід підвищує обґрунтованість архітектурних рішень і знижує ризики на ранніх етапах проєктування програмного забезпечення.

2. Висновок про відповідність проекту поставленому завданню Дипломний проект виконаний відповідно до поставленого завдання та з дотриманням всіх вимог.

3. Характеристика виконання кожного розділу проекту, ступінь використання останніх досягнень науки і техніки та передових методів роботи У першому розділі виконано аналіз предметної області багатокомпонентних програмних систем і сучасних агентно-орієнтованих підходів, що відповідають актуальним досягненням науки і техніки. У другому розділі обґрунтовано концепцію та розроблено метод проєктування на основі багатоагентної взаємодії, використано сучасні підходи колективного прийняття рішень і формалізації знань. Третій розділ присвячено розробці механізму добору мета-моделей із застосуванням кількісного оцінювання та переговорів агентів. У четвертому розділі реалізовано прототип програмного застосунку, проведено тестування, валідацію та аналіз продуктивності з використанням сучасних інженерних інструментів. Робота базується на передових методах агентно-орієнтованого проєктування та підтримки прийняття рішень.

4. Позитивні сторони проекту Тематика дипломного проекту є актуальною, оскільки на робота з багатокомпонентних програмних систем є досить затребуваною.

5. Негативні сторони проекту У проекті опис готових рішень не є повним, доцільно додати розширений аналіз прийнятих рішень.

6. Оцінка графічного оформлення та пояснювальної записки проекту Графічне оформлення виконано відповідно до теми дипломної роботи та подано у вигляді діаграм і рисунків. Пояснювальна записка оформлена згідно вимог чинних стандартів

7. Відгук про дипломний проект в цілому Кваліфікаційна робота заслуговує позитивної оцінки. Матеріал пояснювальної записки структурований, послідовний, чіткий та простий, що дозволяє чітко зрозуміти викладений матеріал у рамках тематики дипломної роботи. Графічний матеріал дає можливість наочно побачити деталі проектування програмного засобу.

8. Інші зауваження

9. Оцінка дипломного проекту Кваліфікаційна робота виконана на достатньому рівні, відповідає поставленій задачі та заслуговує на оцінку «добре».

РЕЦЕНЗЕНТ Тютюнник Єлизавета Тимофіївна, д.т.н., професор, професор каф. КІС

«15» 12 2025 р. _____
(підпис)

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: «Метод проєктування програмного забезпечення на основі агентно-орієнтованої архітектури для багатокомпонентних програмних систем»
Автор: Горбатюк Іван Володимирович
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Науковий керівник: Форкун Юрій Вікторович, кандидат технічних наук, доцент
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) у тексті дипломної роботи системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, бланк завдання), у структурі змісту, назвах розділів/підрозділів тощо та в назвах публікацій у переліку джерел посилання;

2) в якості запозичень системою було зафіксовано деякі послідовності вихідного коду і посиланя на бібліотеки, які є стандартними мовними конструкціями програмування та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) усі запозичення є фрагментарними або мають належним чином оформленні посилання;

4) виявлені модифікації тексту не впливають на відсоток схожості.

Максимальний обсяг запозичень, визначений системою Anti-Plagitarism, складає 2 %. За системою StrickerPlagitarism коефіцієнт подібності складає 1,7 %, коефіцієнт подібності 2 складає 0,7%.

Дата 12.12.25

Керівник _____ Ю.В. Форкун

Гарант ОП _____ О.М. Яшина

Завідувач кафедри _____ Л.П. Бедратюк