УДК
КП
№ держреєстрації 0100V001722
Інв. №

Міністерство освіти України
Запорізький державний університет
(ЗДУ)
69063, м. Запоріжжя, вул. Жуковського, 66; тел. (0612)-64-45-46, факс (0612)-62-71-61, телекс 12-74-47 ОРИОН

ЗАТВЕРДЖУЮ
проректор з наукової роботи
та міжнародного співробітництва,
д-р. техн. наук, проф.

_____В. З. Грищак

ЗВІТ

ПРО НАУКОВО - ДОСЛІДНУ РОБОТУ
Розробка математичних моделей та методів опису і взаємодії елементів єдиного інформаційного простору в інтегрованій мережі Вузів на базі принципів діакоптики і архітектур типу майстер - агент.

ФОРМАЛЬНІ ПРИНЦИПИ І МЕТОДИ ВЗАЄМОДІЇ МОДЕЛЕЙ ФУНКЦІОНАЛЬНИХ ОБ'ЄКТІВ ЄДИНОГО ІНФОРМАЦІЙНОГО ПРОСТОРУ

(проміжний)

Керівник НДР
зав. каф. ММІТ,
д-р. техн. наук., проф. В.О. Толок

2000

Рукопис закінчено 15.11.2000.

ПЕРЕЛІК АВТОРІВ

Керівник НДР
зав. кафедрою
д-р. техн. наук., професор — В.О. Толок
(Вступ, розділи 1,2.3, висновки)

Відповідальний виконавець НДР
ст. наук. співр.
канд. техн. наук, .доцент — С.Ю. Борю
(Вступ, розділи 1,2,3, висновки)

ст. наук. співр.
канд. фіз. - мат. наук, доцент — В.А. Єрмолаєв
(Вступ, розділи 1,2,3, висновки)

наук. співр. — Н. Г. Кеберле
(розділи 1,2)

мол. наук. співр. — С. Л. Плаксін
(розділи 2,3)

мол. наук. співр. — В. В. Міхайліченко
(розділ 2)

Реферат

Звіт НДР: 49 с., 54 джерел

Об'єкт дослідження – інтегровані комп'ютерні мережі і системи, інтегровані розподілені інформаційні системи, інтелектуальні візуальні інтерфейси доступу до компонентів єдиного інформаційного простору, формальні моделі і методи побудови функціональних компонентів єдиного інформаційного простору, мультіагентські системи, архитектуры типу CORBA і майстер - агент, динамічні співтовариства інтелектуальних інформаційних агентів, середовище інтернет і интранет.

Мета роботи - розробка математичних моделей і методів опису і взаємодії уніфікованого інформаційного простору в інтегрованій мережі вузу на базі принципів диакоптики й архитектур типу майстер - агент.

Методи дослідження - математичне моделювання, загальна теорія систем.

Отримані результати і новизна – розроблени формальні принципи і методи взаємодії моделей функціональних об'єктів єдиного інформаційного простору.

Область застосування - наукові, навчальні і дослідницькі роботи.

МАКРОМОДЕЛЮВАННЯ, ІНТЕГРОВАНІ КОМП'ЮТЕРНІ МЕРЕЖІ, ІНТЕГРОВАНІ РОЗПОДІЛЕНІ ІНФОРМАЦІЙНІ СИСТЕМИ, МАТЕМАТИЧНІ МОДЕЛІ, ЄДИНИЙ ІНФОРМАЦІЙНИЙ ПРОСТІР, ДИНАМІЧНІ МУЛЬТИАГЕНТСЬКІ МОДЕЛІ, АРХІТЕКТУРИ МАЙСТЕР-АГЕНТ.

Зміст

Вступ

Основна ідея, що лежить у фундаменті даного дослідження з побудови формальних моделей і методів взаємодії у межах Єдиного Інформаційного Простору (ЄІП) [1], була запропонована в ICDT-моделі Стратегій Бізнесу у Інтернет (Internet Business Strategies) 2. Концепція ЄІП відрізняється від подібної ідеї Віртуального Інформаційного Простору в моделі ICDT, яка визначає простий канал для відображення і доступу до інформації. В рамках нашого дослідження під ЄІП розуміється віртуальний посередник, організований на верхньому рівні багатошарової ІС, що поєднує ієрархію розподілених, неоднорідних, взаємодіючих функціональних компонент (відділів) і розподілені неоднорідні інформаційні ресурси (ІС локального використання). Люди використовують ЄІП як модель ВУ і спілкуються з нею за допомогою уніфікованого візуального Інтранет-інтерфейсу (УВІІ) 1. Концепції ЄІП і УВІІ близькі за змістом до відомих підходів до розробки Населених Інформаційних Просторів 3. В ЄІП мешкають активні функціональні компоненти (мультіагентні системи (МАС) та агенти - учасники), які займають відповідні організаційні клітинки на різних рівнях. З організаційної точки зору ці компоненти є віртуальними бізнес-об'єктами, що виконують бізнес-процеси як потоки робіт, в термінах, наприклад, середовища підприємства 4.

Особливістю пропонованого формального підходу є спроба промоделювати реальні функціональні компоненти ВНЗ і реальні процеси, що виконуються у ВНЗ, як бізнес-процеси в середовище ЄІП. Бізнес-процеси в свою чергу моделюються як процеси інформаційного обміну між різними типами користувачів-людей та різноманітними активними функціональними системами/компонентами, представленими як МАС/агенти, що мають відповідними ролями і розподілені в Інтранет. Середовища, архітектури і їх реалізації для моделювання бізнес-процесів і управління в рамках Віртуального Університету сьогодні набувають значного поширення та дослідження (наприклад, див. [4-5]). Але, та різноманітність процесів, з якою ми зустрічаємось в реальному житті, показує, що дуже важко моделювати їх більш-менш статичними засобами, наприклад, такими, що пропонуються в ROOM, моделях ролей OOFRam 6, середовищі, заснованому на CTL 5, ICRF 7. Парадигма використання агентів, на нашу думку, пропонує вихід із цього світу наперед заданих потоків робіт і специфікацій ролей. Даний підхід використовує метафору динамічних суспільств агентів[1], для того, щоб використовувати кращі засоби для моделювання внутрішньої динаміки Предметної Області. Цей підхід близький до підходу, використаного в середовищі RETSINA 9 для адаптивної взаємодії між командами агентів, що дозволяє вирішувати завдання по прийняттю рішень і управлінням інформацією. В рамках запропонованого підходу агенти є учасниками різних статичних МАС, які представляють постійно існуючі відділи університету. Відділи спілкуються між собою за допомогою так званих агентів-посередників (Proxy Agents), що діють як виконавці, але мають деякі зовнішні функції та додаткові шляхи комунікації. В свою чергу, посередники формують МАС університету більш високого рівня. На нижньому рівні кожний агент-учасник МАС відділу може бути розширений в підпорядковану МАС з тією ж архітектурою. Оскільки ці моделі відділів представляють функціональні відділи університету, вони були розроблені так, щоб "вміти" виконувати належні завдання стосовні свого відділу. До таких завдань належать завдання отримання інформації, інтеграції, обміну і посередництва. Ролі агентів 10 є більш-менш статичні, оскільки агенти здатні виконати задані набори атомарних робіт (політик). З другого боку, здатності агентів змінюються з часом, як змінюються їх накоплений досвід та обмеження, в яких вони працюють. Більш того, агенти в рамках МАС динамічно формують коаліції, які називаються суспільствами агентів, для виконання тієї чи іншої роботи. Запропонований підхід використовує діакоптичне середовище

[1] *Бізнес-процеси розглядаються, наприклад, у Jennings et al 8 в проекті ADEPT як Суспільство агентів , що торгуються.*

МАС 10, модель виконання завдання суспільством агентів 11. Розробка дизайну для користувачів - людей базується на концепції УВІ 12.

Легко бачити, що проблема моделювання віртуального або реального університету засобами ЄІП, в якому мешкають МАС, має два аспекти. З одного боку, це проблема налагодження семантичних, і психологічних зв'язків між виконавцями-людинами та їх штучними дублерами, а також між агентами. З другого боку, це проблема операційній інтероперабельності між неоднорідними розподіленими компонентами - тобто комунікації, спільної роботи і координації між цими учасниками. Проблема ускладнюється, якщо прийняти до уваги аспект еволюції штучних акторів (так само, як відповідних інформаційних ресурсів, представлених у різних моделях даних).

Цій звіт розглядає питання як операційних, так і семантичних аспектів (представлення онтологій, інтероперабельність, спільне використання знань тощо). Операційні аспекти моделювання функціональних компонент розглянути в розділі 1. Проблема взаємовідносин між виконавцями - агентами формально вирішується в розділі 2. Розділ 3 розглядає приклади моделювання двох характерних для ВНЗ бізнес-процесів.

Підставою для виконання даної науково - дослідної роботи є наказ ЗДУ № 60 від 25.02.1999 р.

1 Розробка базових моделей представлення функціональних об'єктів Єдиного Інформаційного Простору

Даний підрозділ презентує формальний підхід, що імплементує функціональні об'єкти у кібер-середовищі, заснованом на агентах, для моделювання процесів інформаційного обміну у Єдиному Інформаційному Просторі (ЄІП) 1, населеном агентами, які створюють суспільства для підтримки виконання різних бізнес-процесів, яки є характерними для вищого навчального закладу (ВНЗ) і віртуального університету (ВУ).

1.1 Базові принципи моделювання функціональних об'єктів в ЄІП

В рамках даної роботи було запропоновано [10,11] середовище для моделювання процесів функціональних взаємозв'язків між розподіленими компонентами (які є інтелектуальними агентами) динамічної багатофункціональної інформаційної системи підприємства / відділу, особливістю якого є той факт, що асоціації між вузлами / акторами є нежорсткими і змінюються із часом. Підхід заснований на діакоптичному принципі [13] і методології [14] макромоделювання. Використання діакоптики дає такі елегантні рішення, як:

- представлення як компонентів системи, так і системи в цілому за допомогою функціонально еквівалентних, об'єднаних і спрощених моделей програмного забезпечення (інтелектуальних агентів із їхніми моделями поведінки, поданими відповідно до макромодельних програм);
- включення топології взаємозв'язків компонентів у загальну макромодель;
- включення спеціалізацій компонентів у загальну макромодель.

Переваги цього середовища такі: з’являється можливість взяти узагальнені моделі к представленню компонентів, їх комунікацій між собою, підсистем і системи як єдиного цілого; є гарні можливості працювати із підсистемами за допомогою більш простих моделей компонентів незалежно від їх внутрішніх рівнів складності; є можливості використати узагальнені архітектури, інтерфейси і реалізовувати масштабовані задачі. Замість створення досить складних, жорстких моделей із сильно зв’язаними компонентами, метод дозволяє оперувати з колекціями "автономних" агентів, які динамічно формують суспільства для виконання того чи іншого процесу інформаційного обміну (завдання). Це зменшує складність, додає руху і дуже підходить для моделювання бізнес-процесів як процесів інформаційного обміну, оскільки пропонує підхід розділення процесу на частини. Набір моделей для представлення функціональних вузлів ЄІП включає: модель компоненти/функціональної системи 10, модель процесу 11, узагальнену модель агента 10.

Актори у середовищі є інтелектуальними (раціональними – у Nwana 15) програмними агентами. Вважається, що завдання, які виконують ці актори – є набірами атомарних робіт. Кожний актор (агент) здатен виконати деяку атомарну роботу із множини дозволених атомарних робіт функціональної системи. Ці здатності формують роль відповідного агента. Нотація ролі, що використана у ЄІП 10, близька до нотації, що прийнята у ICRF 7.

1.2 Модель функціональної системи/компоненти

На рівні функціональної системи були зроблені деякі базові припущення для того, щоб спростити середовище і надати бажаний рівень реактивності агентів. Учасники функціональної

системи вважаються жорстко орієнтованими на командну роботу і "чесну гру". Успішне виконання завдання має більший пріоритет ніж локальні цілі окремих агентів. Агенти, що приєднуються до суспільства, обмежені тим, що мають видавати вірні результати навіть якщо це не узгоджується з їх локальними цілями.

Модель функціональної системи, а також і модель функціонального компонента системи будується на ідеях "узагальнення" і "абсорбції" атомарних робіт із множини дозволених робіт $W = \{w_1, w_2, ...\}$ цієї функціональної системи. Вважається, що сенсорний вхід функціональної компоненти i приймає завдання $W_i \subseteq W$. Деяка частина цього завдання (атомарна робота W_i^p) може бути виконана ("абсорбована") конкретною компонентою, а інші частини завдання або направляються іншим компонентам системи W_i^d, в тому випадку, якщо функціональна компонента знає кому переслати ці роботи, або відхиляється W_i^r Функціональний компонент може створити додатковий набір атомарних робіт W_i^g для завершення виконання роботи W_i^p. Ці роботи W_i^g, разом із W_i^d, далі направляються іншим компонентам:

$$W_i \to F_O^i(W) \to \tilde{W}_i, \tag{1.1}$$

де: $W_i = \{W_i^p, W_i^d, W_i^r\}$, $\tilde{W}_i = \{W_i^d, W_i^g\}$, $F_O^i(W)$ - макромодельна програма.

В спеціальному випадку компонента i може створити новий набір робіт W_i^g, навіть якщо вхідного впливу w_i не було, тобто, компонент може “викликати” нове (під)завдання:

$$F_O^i(W) \to \tilde{W}_i, \tag{1.2}$$

де: $\tilde{W}_i = \{W_i^g\}$, $F_O^i(W)$ - макромодельна програма.

Тому, завданням для кожного функціонального компонента/моделі системи є відповідне виконання (1.1) і (1.2). На даний момент наша модель обмежується таким правилом, що тривалість виконання кожної атомарної роботи $w_j \in W$ - це заданий проміжок часу Δt. Але, вочевидь, це обмеження не є дуже сильним і може бути знято у майбутньому.

1.3 Модель процесу виконання робіт

Функціональна система призначена для виконання процесів. Процес зазначимо як потік виконання завдання. Процес Π_a починається із створення нового завдання $W_a \subseteq W$. Завдання W_a, а також додаткові завдання $\tilde{W}_a$ пов’язані із процесом Π_a і помічені унікальним ідентифікатором цього процесу. Компонент вважається ***пов’язаним з процесом*** Π_a тоді, коли цей компонент може виконати частину завдання W_a, $\tilde{W}_a$, або може створити роботу W_a^g. Агент, що представляє цей функціональний компонент, ***входить до суспільства агентів, виконуючих завдання***.

Процес Π_a вважається виконаним, якщо всі компоненти припинили абсорбувати атомарні роботи всіх завдань, пов’язаних з процесом Π_a. Множина робіт $W_{\Pi_a}^z$, не абсорбована процесом Π_a, помічається як ***множина робіт, що не можуть бути виконані***. Див. Рис. 1.1.

Моделювання процесу Π_a (в усталеному режимі) проводиться із застосуванням схем (1.1) і (1.2) до усіх компонентів системи, до тих пір, як весь процес не виконаний.

Залежність $F_O^i(W)$ моделюється в рамках узагальненої моделі агента 10, або за допомогою будь-якого подібного методу. Вимогою для цієї частини функціональної компоненти є адекватне виконання схем (1.1), (1.2).

На практиці буде розумніше обмежити множину дозволених атомарних робіт, що їх виконує система, а також вважати, що ця множина є скінченною: $W = \{w_1, w_2, ..., w_\sigma\}$. Тоді моделювання роботи функціональної системи у цілому по виконанню завдання організується як дворівневий процес, що виконується поступово, в дискретні проміжки часу t_n, $t_{n+1} = t_n + \Delta t$.

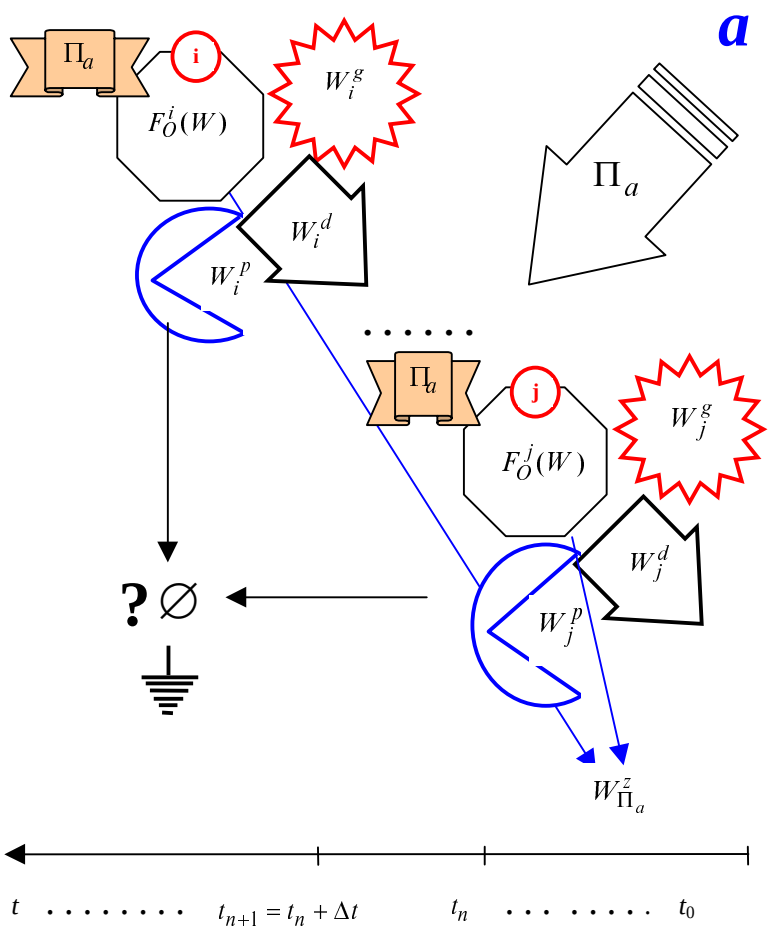

Рисунок 1.1 Модель процесу.

Модель виконання завдання: Нехай $W = \{w_1, w_2, ..., w_\sigma\}$ - множина припустимих атомарних робіт функціональної системи.

На першому (верхньому) етапі виконується збирання станів всіх компонент в об'єднану модель системи станів в період $t_n + \Delta t$.

Ця об'єднана модель має вигляд матриці $\Omega(t_n + \Delta t)$ розмірності $m \times \sigma$, де m - це кількість компонентів системи, а σ - це кількість атомарних робіт в завданні W. Строки матриці Ω (Рис. 1.2) є векторами $\Theta_i = \{k_1, k_2, ..., k_j, ..., k_\sigma\}$, що відображають стани компонентів, де k_j - це стан компонента i при виконанні атомарної роботи w_j. В найпростіших випадках, роль параметра k_j може бути такою:

$k_j = 0$ - компонент зараз виконує атомарну роботу w_j;

$k_j = l > 0$ - компонент зараз виконує атомарну роботу w_j і ще l таких самих робіт чекають у черзі;

$k_j = l < 0$ - компонент був здатен, але не виконав l атомарних робіт w_j (тобто , був "ледачий").

Матриця станів системи $\Omega(t_n + \Delta t)$ формується агентом-координатором із матриць $\mathbf{K}_i$ (розмірності $m \times \sigma$), які представляють стани компонентів. Матриці $\mathbf{K}_i$ створюються виконавцем макромодельної програми $F_O^i(W)$ моделі компоненти на другому етапі моделювання так, щоб забезпечити вхідні дані для формули:

$$\Omega = \sum_{i=1}^{n} \mathbf{K}_i \,. \qquad (1.2)$$

Функціональний компонент може перенаправити собі одну чи більше робіт w_j із завдання W_i для виконання у наступний момент часу. В цьому випадку вектор D_a затримок роботи над процесом Π_a поновлюється так:

$$D_a[j] = D_a[j] + 1 \qquad (1.3)$$

На другому (нижньому) рівні кожна компонента системи (агент) створює $\mathbf{K}_i$. Ці компоненти, як було зазначено вище, моделюються вільно($F_O^i(W)$), але так, щоб вхідна інформація для компоненти i була така: вектор Θ_i, матриця $\Omega(t_n)$ та матриця $\mathbf{K}_i$, означені на попередній момент часу t_n. Матриця $\mathbf{K}_i$ будується згідно з правилом, поданим на Рис. 1.3. і тому відображає поведінку компонента за інтервал часу $[t_n, t_n + \Delta t]$,

де k_{lj}, $l \neq i$: **1** – компонент i віддає роботу w_j компоненту l,

0 – в іншому випадку;

k_{ij}: **-1** – компонент абсорбує (або здатен виконати) роботу w_j за проміжок часу

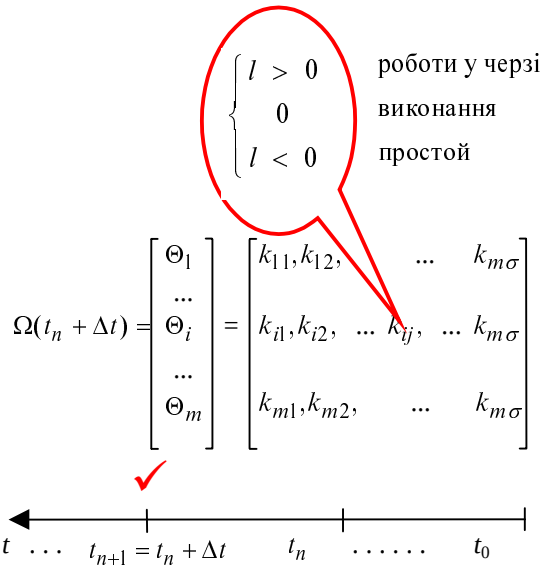

Рисунок 1.2 Стан системи в момент $t_n + \Delta t$

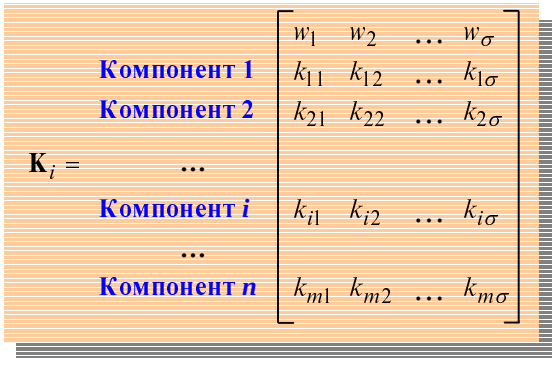

Рисунок 1.3 Можливості і наміри компонентів

$]t_n, t_n + \Delta t]$,

1 – компонент *i* віддає роботу w_j собі,

0 – компонент *i* не може виконати роботу w_j за даний проміжок часу $]t_n, t_n + \Delta t]$.

Аналіз значень $\Omega(t_n)$ може забезпечити уніфіковану оцінку завантаженості компоненти, розподілення "ледачих" станів у кожний інтервал часу $]t_n, t_n + \Delta t]$, і тому розмірковувати про необхідність зміни поведінки компоненти $F_O^i(W)$ в майбутньому.

1.4 Узагальнена модель агента

На рівні агента середовище забезпечує ключові характеристики агента: розміщеність, автономія, раціональність і адаптованість. Агент приймає зовнішній вплив, перевіряє, чи узгоджується цей вплив із роллю агента і чи відповідає його поведінці, і виконує необхідну макромодельну програму – виконує або відхиляє атомарну роботу. Функція макромоделі полягає також у тому, щоб раціонально сформувати відгук, що містить отримані результати. Результати можна представити як функції від параметрів вхідного впливу.

Формально узагальнений агент є реактивним, раціональним, складається із сенсорного інтерфейсу, що сприймає зовнішній вплив, з каскаду із трьох скінченно-станових машин для верифікації вхідного впливу, локальної бази знань і блоку виконання макромоделі. Узагальнений агент є операційною оболонкою, що забезпечує кістяк будь-якого агента середовища. Агенти спеціалізуються на базі множин макромодельних програм, що відповідають їхнім ролям. Макромодельні програми вважаються політиками агентів і зберігаються у її локальних базах знань.

Зроблено припущення, яке обгрунтовується прикладами, що приведени в розділі 3 даного звіту: кожний функціональний елемент (агент), діє по слідуючому узагальненому алгоритму:

```
DO    While .T.
      A = Wait(Action)                    //Среагувати на вхідний вплив
      Ỹ = Process(ActionEvent)            //Обробити отриманий вплив
      IF (Requested)
            SendResults(A, Ỹ)             //Возвернути реакцію на вплив
      END IF
LOOP
```

где:

- Wait() - процедура чекання події впливу.
- Process() - процедура обробки отриманого впливу, що виконує наступну послідовність дій:

  Перевірити свої повноваження (чи є право реагувати на отриманий вплив $a(f, X, Y)$ у поточному стані s_i).

  Перевірити відповідність отриманих параметрів $X = (x_1, ..., x_n)$ політиці f і граничним умовам стану s_i.

  Перевірити формальну відповідність вектора результату $Y = (y_1(X), ..., y_m(X))$ політиці f і стану s_i.

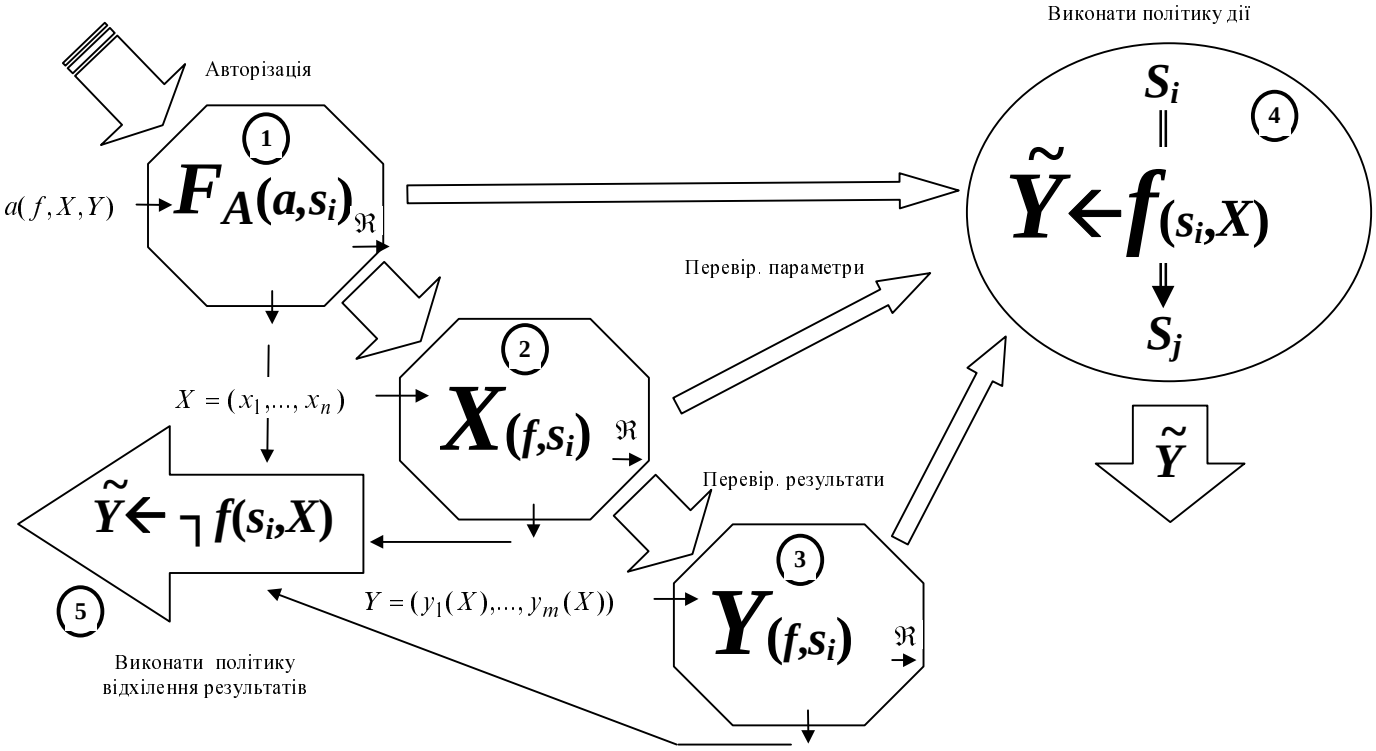

Рисунок 1.4 Формальна узагальнена схема агента, що оброблює вплив.

Виконати політику f, сформувати вектор результатів $\tilde{Y} = (\tilde{y}_1(X),...,\tilde{y}_k(X))$, змінити стан $s_i \rightarrow s_j$.

- SendResults() - процедура повернення реакції на оброблений вплив.

Структурна схема інтелектуального агента, що моделює поведінку функціонального елемента суспільства, приведена на Рис. 1.4. Кожний із блоків **1 - 3** являє собою кінцевий автомат, на вхід якого подаються:

- для блока **1** - ланцюжок «символів»*, що описує зовнішній вплив;
- для блока **2** - ланцюжок «символів», що описує вхідні параметри зовнішнього впливу;
- для блока **3** - ланцюжок «символів», що описує очікувані результати, що об'єкт, що робить вплив на наш агент, розраховує одержати в результаті виконання запитаної політики *f*.

У залежності від того, у яке зі станів переходить кожний із кінцевих автоматів у результаті розбору вхідного ланцюжка, агентом приймається та або інша стратегія поведінки. Так при переході одним з автоматів 1-3 стан, що відхиляє вхідний ланцюжок, формується результат, що говорить про те, що даний агент A у даному стані s_i не може виконати політику f по одній із наступних причин:

- агент A в даному стані s_i не авторизований для виконання політики f - стан блока, що **відхиляє -- 1**;
- вхідні параметри $X = (x_1,..., x_n)$ не відповідають системним параметрам агента А, або не відповідають обмеженням на системні параметри агента A, що накладається станом s_i -- стан блока, що **відхиляє -- 2**;

---

* - *під «символом» ми будемо розуміти елемент вхідного алфавіту* $\aleph$ *відповідного кінцевого автомата*

- описи очікуваних результатів $Y = (y_1(X),..., y_m(X))$ не відповідають тим результатам, що може одержати агент А виконуючи політику f (наприклад, $y_1(X)$ містить ім'я файла, а $\tilde{y}_1(X)$ в результаті виконання політики f одержить значення, що представляє собою вектор (0, 0.25, 0.5, 0.75, 1)) - стан блока, що **відхиляє -- 3.**

Блок **4** являє собою виконавець, у функцію якого входить виконання політики *f*, запитаної зовнішнім впливом *a*. Зі схеми, приведеної на Рис. 1.4, випливає, що цей блок буде виконуватися тільки тоді, коли кожний із кінцевих автоматів **1 - 3** у результаті аналізу своїх вхідних ланцюжків перейде в стан , що дозволяє. Завданням блока **4** є виконання політики *f*, формування результату $\tilde{Y} = (\tilde{y}_1(X),..., \tilde{y}_k(X))$ і перевід агента *A*, у разі потреби, із стана s_i у стан s_j.

Блок **5** є тією частиною моделі, що формує відхиляючу реакцію агента А на *вплив a*. реакція агента, що відхиляє вплив, залежить від від *політики f*, запитаної впливом а, стану s_i, *параметрів* X, Y і далеко не завжди **є фактично негативною**.

Виходячи з вищевикладеного, формальна модель агента може бути подана в такий спосіб:

$$A = \{F_a, F_X, F_Y, S, X_A, F, F_O\}, \qquad (1.4)$$

де $S_A = \{s_1,..., s_n\}$ - множина станів агента *A*.

$X_A = \{x_1,..., x_p\}$ - множина системних параметрів агента *A*.

$F_a = F_a(a, F, s_i) = \{\aleph_a, \wp_a(s_i), \Re_a, \delta_a(s_i)\}$ - кінцевий автомат, що здійснює авторизацію політики зовнішнього впливу a в стані $s_i \in S$. Вхідна послідовність а записується в «символах» вхідного алфавіту $\aleph_a$. Множина станів кінцевого автомата $\wp_a$ є функція від стана s_i агента A и елементи множини станів залежать від стану s_i агента A. Множина станів , що $\Re_a$ дозволяють, складається з одного елемента - стану, що позначений на Рис.1.4. символом $\Re$. Функція переходів δ також залежить від стану s_i агента *A*.

$F_X = F_X(X, X_A, s_i) = \{\aleph_X, \wp_X(X_A, s_i), \Re_X, \delta_X(X_A, s_i)\}$ - кінцевий автомат, що здійснює перевірку відповідності параметрів X отриманого впливу а на відповідність множини системних параметрів X_A і обмеженням стану s_i.

$F_Y = F_Y(Y, f, s_i) = \{\aleph_Y(f), \wp_Y(f, s_i), \Re_Y(f), \delta_Y(f, s_i)\}$ - кінцевий автомат, що здійснює перевірку відповідності опису очікуваних результатів Y тим результатам, що можуть бути отримані при виконанні агентом A в стані *si* політики f.

$F = \{f_1,..., f_i,..., f_m\}$ - роль агента - множина політик, виконуваних агентом *A*.

F_O - програмна компонента, що виконує політику *f*.

$f \in F$ - політика дії - набір атомарних інструкцій, поданий у вигляді програми, що містить розгалуження. Подібні програми описані, наприклад у [16]. Структура політики дії по нашому уявленню виглядає в такий спосіб:

```
DO CASE
        CASE State = s1
        StartThread(<Id1>)
        BEGIN
            Execute <Instruction> on (X̃i) results to (Ỹi)
            Broadcast <Action> with (X̃i, Ỹi) and wait for (Ỹi)
                                        from each party
```

```
                OptimiseResults()
                Require Aj  with (< Action >, X̃i, Ỹi) and wait for (Ỹi)
                Invoke Aj  with (< Action >, X̃i, Ỹi)
                ChangeState(s1)
            END
            CommitThread(<Id1>)
            .....
            StartThread(<Idk>)
            BEGIN
                        .....
            END
            CommitThread(<Idk>)
        ..........
            CASE State = sn
            ..........
            OTHERWISE
            ..........
    END CASE
```

де $<Id_i>$ - ідентіфікатор потока

<Instruction> - інструкція, що виконує атомарну дію в межах даній політики без залучення інших агентів

<Action> - ідентифікатор впливу, що визначає частину політики дії, що не може бути виконана даним агентом. Для впливу *<Action>* може бути початий один із наступних методів виконання:

Broadcast *<Action>* для випадку, коли авторізований для *<Action>* агент не відомий. У цьому випадку вплив *<Action>* робиться на всі члени співтовариства.

Require *Aj* with $(< Action >, \tilde{X}_i, \tilde{Y}_i)$ для випадку, коли авторізований для *<Action>* агент відомий. Цей спосіб моделює випадок синхронної взаємодії з агентом *Aj*. Результати, повернуті агентом *Aj*, складуть частину результатів виконуваній політики.

Invoke *Aj* with $(< Action >, \tilde{X}_i, \tilde{Y}_i)$ для випадку, коли авторізований для *<Action>* агент відомий і нам не важлива його реакція на вплив *<Action>*. У цьому випадку вектор значень, що повертаються, вірогіднійше усього буде порожнім. Інтерфейс примітива **Invoke** проте містить параметр $\tilde{Y}_i$ на той випадок, якщо в майбутньому буде реалізований механізм напрямку результатів третій стороні A_m.

$$X = \bigcup_i \tilde{X}_i, \quad Y = \bigcup_i \tilde{Y}_i$$

Такі впливи реалізуються за допомогою механізму комунікації агентів, формальна модель якого доводиться в розділі 2 цього звіту.

1.5 Висновки

Розробка моделей функціональних об'єктів і методів їхньої взаємодії доцільна на основі застосування диакоптичного підходу для математичного моделювання.

Запропоновані вище математичні моделі агентів і завдань, які вони виконують, дозволяють організувати процес моделювання практично довільної функціональної системи, опис якої можливо здійснювати в термінах «завдання», «робота» і т.і.

2 Розробка формальних принципів і моделей взаємодії функціональних об'єктів ЄІП

Функціональна проекція VIS населена МАС, що представляють функціональні системи і компоненти на різних рівнях. Агенти – учасники динамічно формують проблемно-орієнтовні суспільства для виконання завдань отримання інформації, інтеграції, посередництва і обміну, що з'являються у рамках МАС на одному чи іншому рівні.

В реальному житті такі завдання розділяються і виконуються групами функціональних компонентів. Традиційні моделі процесів інформаційного обміну часто базуються на структурованих наборах жорстких відношень між функціональними вузлами. При моделюванні з використанням більшості середовищ вищеназвані завдання (потоки робіт) представляють як плани – наперед задані орієнтовані графи, мережі Петрі 17 та ін. Такі підходи часто є надто статичними. Наприклад, розглянемо процес обговорювання результатів лабораторних робіт (з курсу хімії). Такий процес не завжди можна промоделювати у вигляді ієрархії жорстко розміщених акторів. Люди часто використовують більш м'які відносини: мозковий штурм, неформальна дискусія, тощо. Друга вада традиційних підходів це то, що жорсткі моделі відношень не є масштабованими. Велика перевага моделі виконання плану 11, що використана у нашому середовищі, – це відсутність статичних наперед заданих специфікацій завдань.

Завдання, в рамках нашого підходу, "викликаються" агентом-посередником і виконуються агентами середнього рівня[2]. Агенти середнього рівня динамічно формують суспільства для виконання наявних завдань. Агент приєднується до суспільства тоді і тільки тоді, коли він сприймає вхідні дані, що містять (під)множину атомарних робіт (частину завдання) для виконання. План виконання завдання уточнюється в подробицях під час процесу покрокового виконання завдання. Процес впроваджується командою агентів-учасників, діючих у кооперації один з одним. Агент-координатор виконує функції координації дії команди агентів і відображає діяльність кожної команди. Агенти – учасники суспільства діють як моделі функціональних компонентів відповідних об'єктів реального світу.

Ключовими проблемами в організації взаємодії агентів, як автономних сущностей, є реалізація засобів передачи впливів і сообщеній (комунікації), а також узгодження спільних робіт (координації) між агентами що діють спільно для вирішення спільної проблеми, або виконання спільного завдання.

2.1 Принципи і методи організації взаємодії агентів

Взаємодія – одна з найважливіших характеристик агентів. Агенти взаємодіють для того, щоб спільно використовувати інформацію, виконувати завдання, досягати спільних цілей.

Гензерет і Кетчпел [19] стверджують, що здатність зв'язуватися на мові взаємодії агентів (ACL) – єдина характеристика, яка відрізняє агентів від іншого програмного забезпечення. Механізми зв'язку накладають додаткові обмеження на внутрішню структуру агенту, яка повинна їх ефективно використовувати [19].

В [20] виділється три типи взаємодії, які можуть використовуватися в МАС:

– Непрямий обмін повідомленнями (indirect message passing);

---

[2] *Агенти середнього рівня в рамках даного підзоду розуміються як такі, що приймають участь у виконанні завдань зсередини МАС відділу і не мають прямого зв'язку із зовнішнім світом. Використана нотація близька до нотації, прийнятої у середивищі RETSINA 18.*

– Пряма взаємодія, із використанням API або Remote Procedure Call (RPC);
– Використання загальнодоступної пам’яті, наприклад, дошку оголошень.

Вони стверджують, що взаємодія агентів за допомогою обміну повідомленнями буде більш гнучкою, ніж загальнодоступна пам’ять (із проблемами паралельного доступу і семафорами) і RPC (який обмежує гнучкість під час виконання), тому що вона дозволяє агентам вільно зв’язуватися і бути розподіленими. Крім того, цей вид взаємодії забезпечує гнучкість ліній зв’язку.

2.1.1 Засоби комунікації агентів

До засобів взаємодії агентів відносять мови взаємодії агентів (FIPA ACL, KQML).

Мова взаємодії агентів забезпечує обмін знаннями і інформацією між агентами. FIPA ACL, на відміну від таких засобів як RPC, RMI, CORBA, що забезпечують обмін інформацією між прикладними задачами, має більш складну семантику [21]. Крім того, ACL має такі переваги в порівнянні із CORBA[22]:
– FIPA ACL управляє судженнями, правилами, діями, а ні семантично не зв’язаними об’єктами.
– Повідомлення FIPA ACL описує очікуваний стан, а не процедуру або метод.

Однак ACL не охоплює повного спектру об'єктів, якими могли б обмінюватися агенти, наприклад плани, цілі, досвід, стратегії.

На технічному рівні, при використанні ACL, агенти транспортують повідомлення по мережі, використовуючи протоколи нижчого рівня, наприклад SMTP, TCP/IP, POP3, або HTTP.

Мова взаємодії агентів (ACL) повинна дозволяти передавати інформацію будь-якого виду між різними агентами. Є два підходи до проектування мов взаємодії агентів:
– ***Процедурний***, включає обмін процедурними директивами/командами. Це може бути реалізовано за допомогою таких мови програмування як Java або Tcl.
– ***Декларативний***, де зв'язок заснований на декларативних інструкціях, типу визначень, припущень, знань, і т.п.

Через обмеження на процедурні підходи (наприклад, такі сценарії важко координувати, об'єднувати), були використані декларативні мови для створення мов взаємодії агентів. Одними з найбільш популярних декларативних мов є KQML [23] із своїми діалектами і FIPA ACL [19].

2.1.2 Методи координації в МАС

Як помічає К.Сікара [9], вивчення мультиагентних систем зосереджується на системах, в яких багато інтелектуальних агентів взаємодіють один із одним. Агенти, як вважається, є автономними об’єктами, типу програм або роботів. Їх взаємодія можуть бути або кооперативними, або егоїстичними. Тобто, агенти можуть разом досягати однієї мети (як у колонії мурашів), або вони можуть переслідувати власні інтереси (як у вільній ринковій економіці). Тому необхідно реалізовувати кожного мураша в колонії так, щоб примусити їх разом приносити їжу найбільш ефективним способом, або встановити такі правила взаємодії, щоб група егоїстичних агентів працювала разом для виконання конкретного завдання.

Таким чином, координація - центральна проблема в МАС зокрема, й у розподіленому штучному інтелекті (DAI) взагалі. Власне кажучи, координація - *процес,* у якому агенти беруть участь, щоб забезпечити суспільству несуперечливу, погоджену послідовність дій. Це означає, що дії агентів

сплановані так, що вони не конфліктують один з одним, а все суспільство поводиться як єдине ціле. Є декілька причин, по котрим МАС повинна бути скоординована [23]:

- *Запобігання анархії або хаосу*: координація необхідна або бажана, тому що в децентралізованої MAS може легко встановитися анархія;
- *Встановлення глобальних обмежень*: завжди існують глобальні обмеження, що МАС повинна задовольнити, щоб успішно виконати завдання;
- *Розподілені знання, ресурси, інформація*: практично завжди в МАС є ресурси, робота з якими вимагає координації;
- Залежність між діями агентів: цілі агентів часто взаємозалежні.

Ефективність: навіть коли агенти можуть працювати незалежно, координація дозволяє заощадити час. Існує багато підходів забезпечення координації в МАС. Усі ці підходи можна розділити на чотири категорії [24]:

- організаційне структурування;
- укладання контракту;
- планування діяльності MAS;
- переговори.

Організаційне структурування: Це найпростіший метод координації, що складається із визначених і довгострокових відношень між агентами (див. огляд [24]) . При цьому часто використовують ієрархічні структури *master-slave* або *client-server*. Ця методика використовується в двох варіантах:

- Головний агент планує і розподіляє завдання (роботи) між підпорядкованими агентами. Підпорядковані агенти, в решті-решт, повинні повідомити головному агенту про результати їхньої роботи. Крім того, на відміну від головного агента, підпорядковані агенти мають часткову автономність.
- Використання класичної дошки оголошень - загального інформаційного поля - для забезпечення координації. У цій схемі агенти використовують дошку для обміну інформацією. Головний агент (планувальник) призначає агентам операції читання/запису з дошкою. Ця схема використовується Веркманом у його DFI системі (див. огляд [24]). Цей підхід може використовуватися, коли завдання розподілене, є центральний агент планування або коли завдання вже були призначені, *апріорно,* агентам. Sharp Multi-Agent Kernel (SMAK) (див. огляд [24]) теж використовує цю стратегію.

Останній варіант показує, що координація в організаційному структуруванні не завжди зв'язана з ієрархією. Наприклад, у системі DVMT (див. огляд [23]), що використовує технологію дошки оголошень, координація проходить серед рівних агентів.

Вузьким місцем у системах, заснованих на технології дошки без прямої взаємодії агентів, може бути велика кількість агентів у суспільстві, навіть у випадку дошки, розділеної на секції. Крім того, всі агенти повинні мати загальне уявлення про дошку. Тому більшість систем, заснованих на цій технології, використовують невеликих гомогенних агентів (як, наприклад DVMT).

Дурфи (див. огляд [24]) вважає, що таке централізоване управління, як в ієрархічній методиці, суперечить основним припущенням про DAI. У цій методиці передбачається, що, принаймні, один агент має глобальне представлення всього суспільства, однак у багатьох областях це не реально. Якщо все-таки ця методика координації використовується, проектувальник повинен гарантувати, що всі підпорядковані агенти мають оптимальний ступінь деталізації щоб час, витрачений на декомпозицію загального завдання на більш дрібні, не перевищило часу її виконання одним агентом.

Укладання контракту: У цьому підході, що припускає децентралізовану структуру, агенти приймають дві ролі:

- менеджер, що поділяє вхідне завдання на підзадачі і шукає виконавця, щоб виконати їх;

– виконавець, що виконує підзадачу. Виконавець може рекурсивно стати менеджером і розбити вхідну підзадачу на ще більш дрібні завдання і доручити їх іншим агентам.

Пошук виконавця виглядає у такий спосіб (FIPA CNP [19]):

Менеджер повідомляє про наявність вхідного завдання;

Виконавці оцінюють це завдання з погляду можливості її виконати;

Менеджер одержує таблицю виконавців;

Оцінює отримані пропозиції, вибирає виконавця і доручає виконати йому завдання;

Чекає результатів виконання завдання.

Це цілком розподілена схема, в якій кожний агент може бути як менеджером, так і виконавцем. Цей підхід застосовується в багатьох прикладних задачах.

Хухнс і Сингх (див. огляд [24]) звертають увагу на те, що ця модель координації забезпечує розподіл загального завдання, і засоби для самоорганізації групи агентів, і найкраще застосовна коли:

– прикладне завдання має чіткий ієрархічний характер (природу);
– прикладне завдання ділене на великі підзадачи;
– є мінімальний взаємозв'язок серед підзадач.

Переваги цього підходу координації є такими: динамічний розподіл завдання завдяки пошуку оптимального виконавця; збалансоване завантаження агентів (зайняті агенти можуть не брати участь у виконанні загального завдання); і надійний механізм для розподіленого керування і відновлення результатів при збої.

Агенти в такій системі досить пасивні і не застосовні для багатьох прикладних завдань. Нарешті, таке суспільство агентів досить інтенсивно використовує мережа, що може знизити усі його переваги.

Планування діяльності MAC: Є два типи планування діяльності MAC:

– централізоване планування діяльності MAC;
– розподілене планування діяльності MAC.

У централізованому плануванні діяльності MAC звичайно є координаційний агент, що, отримавши всі індивідуальні плани від інших агентів, аналізує їх, щоб виявити потенційні протиріччя і конфлікти у взаємодії агентів (наприклад, конфлікт між агентами по використанню загального ресурсу). Потім координаційний агент намагається змінити ці індивідуальні плани й об'єднує їх у план MAC, в якому суперечливі взаємодії усунуті. У заключному плані MAC, додані команди зв'язку, щоб синхронізувати взаємодії агентів у потенційно можливих конфліктах.

У розподіленому плануванні діяльності MAC, ідея полягає в тому, щоб забезпечити кожного агента моделлю планів інших агентів. Агенти взаємодіють, щоб створювати і модифікувати їхні індивідуальні плани, що не конфліктують із планами інших агентів.

Координація в розподіленому плануванні діяльності MAC - набагато складніше, чим у централізованій формі, тому що там не існує агентів, що володіють глобальним представленням про всю розподілену систему.

Переговори: Це один із самих складних методів координації. Будь-яка форма взаємодії між людьми вимагає деякою мірою явних або неявних переговорів. Тому, не вражає той факт, що багато дослідників переговорів як методу координації застосовують саме людські стратегії ведення переговорів. Крім того, при координації агентів цим підходом часто використовують різні методи ИИ, логіки, випадково заснованого міркування, спрямованого обмеженням пошуку, і т.д.

Важливим завданням цього розділу звіту є розробка моделей комунікації і координації агентів, що формують суспільства і виконують завдання (моделюють діяльность реальних функціональних

компонент і систем ВНЗ) в ЄІП. Функціональна проекція ЄІП, як вже зазначалося, населена МАС, що представляють функціональні системи і компоненти на різних рівнях. Агенти – учасники динамічно формують проблемно-орієнтовані суспільства для виконання завдань одержання інформації, інтеграції, посередництва й обміну, що з'являються у рамках МАС на однім або другому рівні.

Велика перевага моделі виконання плану 11 використана в нашому середовищі – це відсутність статичних наперед заданих специфікацій завдань.

Завдання, у рамках нашого підходу, "викликаються" агентом-посередником і виконуються агентами середнього рівня[3]. Агенти середнього рівня динамічно формують суспільства для виконання завдань. Агент приєднується до суспільства тоді і тільки тоді, коли він сприймає вхідні дані, що містять (під)множину атомарних робіт (частина завдання) для виконання. Агенти обмінюються впливами (завданнями) як передбачено моделлю комунікації (див. п. 2.2.). План виконання завдання уточнюється в подробицях під час процесу покрокового виконання завдання. Процес впроваджується командою агентів-учасників, що діють у кооперації один з одним. Агент-координатор виконує функції координації дії команди агентів і відображає діяльність кожної команди. Модель координації приведена в п. 2.3. Агенти – учасники суспільства діють як моделі функціональних компонентів відповідних об'єктів реального світу.

2.2 Модель комунікації агентів, що моделюють функціональні компоненти ЄІП

На рівні комунікації вважається, що агенти, які приймають участь в процесі виконання завдання, спілкуються за допомогою таких актів спілкування:

- *директива* – безумовне виконання дії підпорядкованим агентом (Рис. 2.1).
- *детермінований запит із детермінованою реакцією* – посилаючи повідомлення агент хоче, що б йому повернули деякі результати (Рис. 2.2).
- *детермінований запит із оптимізацією результату* – після одержання результатів запиту, агент, перед використанням результатів, повинний їх спочатку оптимізувати (Рис. 2.3).
- *едетермінований запит із оптимізацією результату* – використовується, коли заздалегідь не відомий агент-одержувач повідомлення. Після того як повідомлення буде послано всім агентам усередині системи й отримані від кожного такого агента результати, буде обрано кращого агента - виконавця (Рис. 2.4).

Якщо представити зовнішній вплив на суспільство і реакцію агента на цей вплив у вигляді наступного співвідношення:

$$\tilde{Y} = a(f, X, Y) \qquad (2.1)$$

де: $a(f, X, Y)$ - вплив, f - політика, $X = (x_1, ..., x_n)$ - параметри,

$Y = (y_1(X), ... y_m(X))$ - опис запитуваних результатів,

$\tilde{Y} = (\tilde{y}_1(X), ..., \tilde{y}_k(X))$ - реакція (результати)

і, крім того, взаємодія агентів усередині співтовариства, у виді:

$$\tilde{Y}^* = a^*(f^*, X^*, Y^*) \qquad (2.2)$$

то:

---

[3] *Агенти середнього рівня в рамках даного підзоду розуміються як такі, що приймають участь у виконанні завдань зсередини МАС відділу і не мають прямого зв'язку із зовнішнім світом. Використана нотація близька до нотації, прийнятої у середивищі RETSINA 18.*

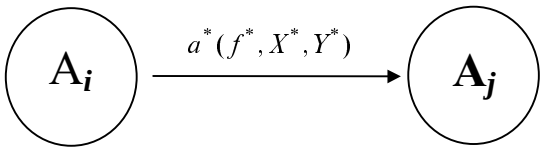

Рисунок 2.1 Директива

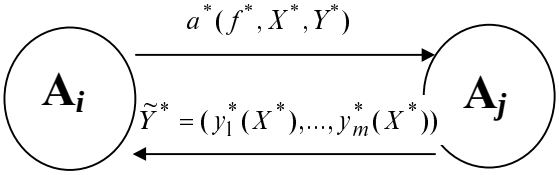

Рисунок 2.2 Детермінований запит із детермінованою реакцією

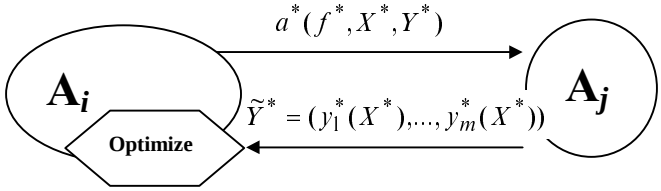

Рисунок 2.3 Детермінований запит із оптимизацією результата

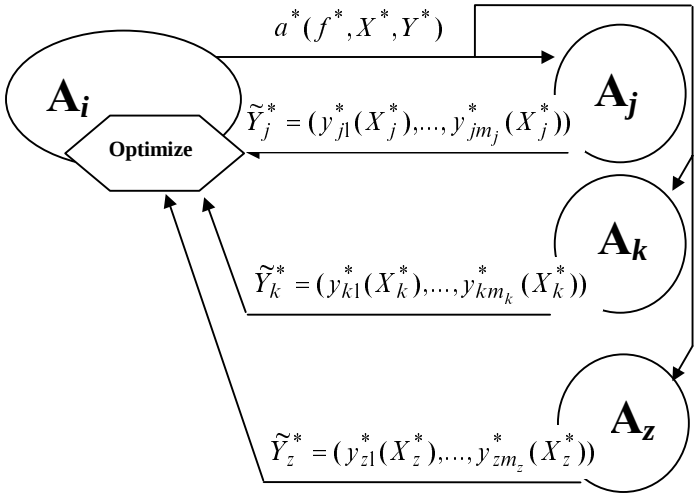

Рисунок 2.4 Недетермінований запит із оптимизацією результатів

для впливу виду «директива» буде справедливо:

$$X^* \subseteq X\ , \qquad Y^* = \tilde{Y}^* = \emptyset, \qquad f^* \subseteq f\ ; \tag{2.3}$$

для впливу виду «детермінований запит із детермінованою реакцією» буде вірно:

$$X^* \subseteq X\ , \qquad Y^* = \tilde{Y}^*, \qquad f^* \subseteq f\ ; \tag{2.4}$$

для впливу виду «детермінований запит із оптимізацією результату» буде вірно:

$$X^* \subseteq X\ , \qquad Y^* \subseteq \tilde{Y}^*, \qquad f^* \subseteq f\ ; \tag{2.5}$$

для впливу виду «недетермінований запит з оптимізацією результату» буде вірно:

$$X^* = X\ , \qquad Y^* = \tilde{Y}^*, \qquad f^* \equiv f\ ; \tag{2.6}$$

Наведені рівняння дозволяють описати будь-який вплив на програмного агента в мультиагентной системі, що виконує завдання (див розділ 1).

2.3 Модель координації виконання множин атомарних робіт

Використовуючи результати першого розділу, диакоптичний підхід до моделювання суспільств агентів, що описаний у [10,11], побудуємо модель координації агентів, засновану на активній дошці оголошень.

Використання активної дошки оголошень відрізняється від класичного підходу координації лише тим, що всі операції читання/запису з загальним полем пам'яті робить тільки один агент - координаційний агент, що, по суті, інкапсулює у самому собі дошку оголошень, і забезпечує методи роботи з нею. Інші агенти, при необхідності прочитати або записати на дошку деяку інформацію, повинні взаємодіяти з координаційним агентом за допомогою ACL/KQML повідомлень.

Щоб більш докладно описати модель координації, послідовність дій агентів, взаємодію агентів суспільства з координаційним агентом, повернемося до приклада з підрозділа 3.1. [25].

Для простоти розглянемо лише процес виконання агентом **PM** роботи w_2 (див. Рис. 2.5).

Робота w_2 надходить до агента **PM** у момент часу t_1. Однак вона не може бути виконана за час $]t_1, t_1 + \Delta t]$, тому що для її виконання необхідний результат роботи W_6 агентом **DBA,** що потрібно запросити у координаційного агента - **COA**. Макромодель $F_O^{PM}(w_2)$ виконання роботи w_2 агентом **PM** може бути подана наступним алгоритмом:

```
  ...
  Avail := Require("COA" with ("ProvideResults", Ỹ(w6)))
 IF .NOT. Avail THEN
     Redirect w2 TO PM   // w2 перенаправляється PM на
                         //наступний момент часу t1 + Δt.
       ELSE
     Execute( w2 )          //виконання w2 агентом PM (див. Рис. 2.5)
     END IF
...
```

У цей же момент часу t_1 до агента **DBA** надходить робота w_6. Агент **DBA** виконує роботу w_6 в інтервалі часу $]t_1, t_1 + \Delta t]$ і відсилає результат $\tilde{Y}(w_6)$ координаційному агенту. Це може бути описане в такий спосіб:

```
  ...
     Ỹ := Execute( W6 )                    //виконання роботи W6
     Invoke("COA" with ("AcceptResults", Ỹ ))   //відправлення
                                                //Ỹ агенту COA
  ...
```

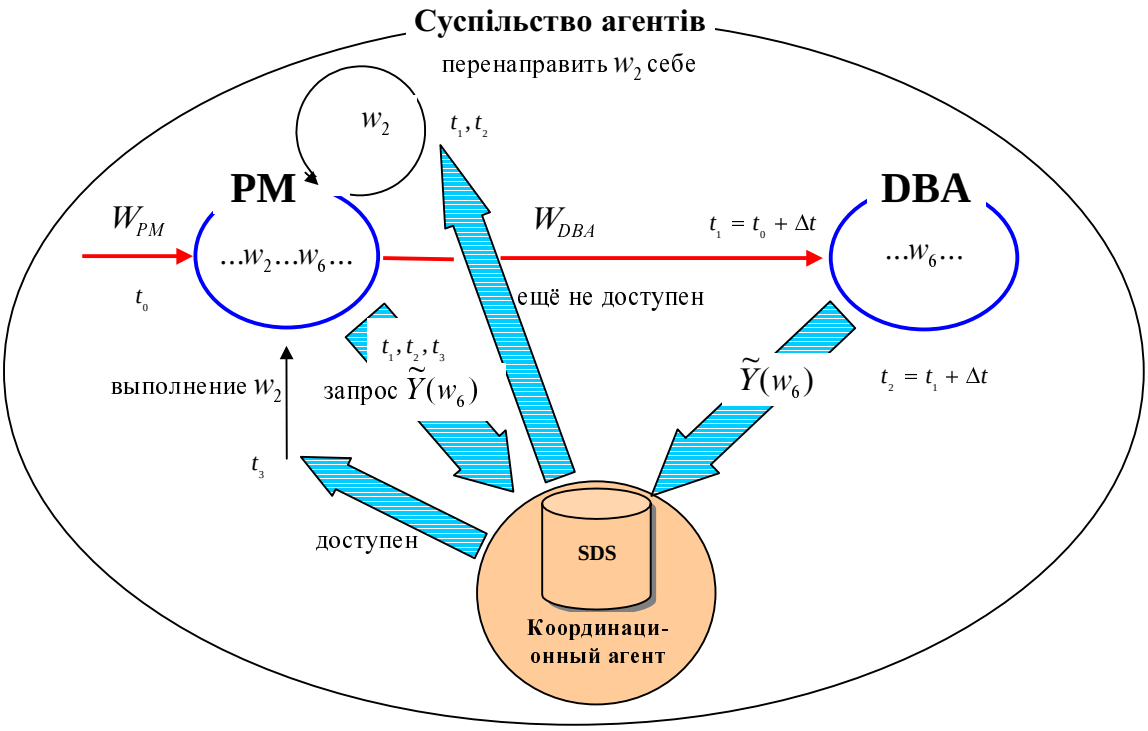

Рисунок 2.5 - Модель координації

Реакція координаційного агента **COA** на отримане від агента **DBA** вплив типу **AcceptResults** має такий вид:

```
...
   Execute("AcceptResults", DBA, Ỹ )
...
```

Реакція **COA** у відповідь на запит **PM** результатів роботи w_6, може бути описана у виді наступного алгоритму:

```
...
 Avail := Execute("ProvideResults", Ỹ(w6)))
IF .NOT. Avail THEN
   Reply("PM"  with  ("AcceptResults", "DUMMY"))
         ELSE
   Reply("PM"  with  ("AcceptResults", Ỹ(w6)))
 END IF
...
```

Таким чином, у момент часу t_2 реакція **COA** буде відхилена, тому що результатів роботи w_6 ще ні, а в момент часу t_3, запит агента **PM** буде задоволений.

Необхідно відзначити, що результати виконаних робіт, що передаються між агентами, повинні бути подані в деякому універсальному форматі, «зрозумілому» всім агентам. Таким форматом може бути KIF (Knowledge Interchange Format) [26] - формат обміну знаннями - це формальна мова

для обміну знаннями між несумісними комп'ютерними програмами, написаними різними програмістами, у різний час, на різних мовах і т.д.

Повертаючись до приклада з [25], запишемо на мові KQML[23] повідомлення, що забезпечують взаємодію з координаційним агентом.

KQML-повідомлення для відсилання результатів виконаної роботи координаційному агенту може мати такий вид:

```
(ask-one
        :sender         "DBA"
        :receiver       "COA"
        :in-reply-to    Null
        :reply-with     Null
        :language       KIF
        :ontology       COA ontology
        :contents       ("AcceptResults", Ỹ(w6) )   )
```

KQML повідомлення для запиту результатів роботи у координаційного агента:

```
(ask-one
        :sender         "PM"
        :receiver       "COA"
        :in-reply-to    Null
        :reply-with     Ỹ(w6)
        :language       KIF
        :ontology       COA ontology
        :contents       ("ProvideResults", Ỹ(w6) )   )
```

KQML повідомлення - повернення результатів координаційним агентом:

```
(tell
        :sender         "COA"
        :receiver       "PM"
        :in-reply-to    Null
        :reply-with     Null
        :language       KIF
        :ontology       COA ontology
        :contents       ("AcceptResults", Ỹ(w6) )   )
```

Таким чином, основними функціями координаційного агента є одержати, деякий час зберігати й відсилати по запиті проміжні результати виконання робіт у співтоваристві.

З огляду на головну функцію координаційного агента в співтоваристві - зберігати проміжні результати робіт, можна сказати, що основними припустимими типами зовнішніх впливів, на які

повинний реагувати координаційний агент є: 1) одержання результату деякої роботи; 2) запит результату деякої роботи.

Далі в цій роботі одержання координаційним агентом результату деякої роботи будемо називати *зовнішнім впливом першого типу*, а запит у координаційного агента результатів деякої роботи - *зовнішнім впливом другого типу*.

У термінах моделі комунікації [10,11], див. підрозділ 2.2., зовнішній вплив першого типу буде директивою, тому що від координаційного агента потрібно лише прийняти результат; зовнішній вплив другого типу буде детермінованим запитом із детермінованою реакцією, тому що координаційний агент повинний повернути запитуваний результат або повідомити, що такого результату ще немає.

Очевидним є той факт, що ефективність роботи всього співтовариства залежить від того, наскільки швидко координаційний агент буде реагувати на зовнішні впливи. Іншими словами критичним параметром роботи координаційного агента є час реакції. Цей висновок дозволяє накласти деякі обмеження на архітектуру самого координаційного агента; на макромоделі, що описують реакцію агента на зовнішні впливи; на організацію збереження даних - результатів робіт. Тому наступні підрозділи другого розділу будуть присвячені цим проблемам.

2.4 Модель еволюції функціональних компонентів в МАС

Одна із найголовніших характеристик деякої функціональної компоненти (і ЄІП у цілому) - це його здатність змінюватися. Середовище для моделювання ЄІП повинно мати засоби для роботи із змінами, що з'являються у реальному світі – треба використовувати моделі, що здатні підтримувати різні типи еволюції. С точку зору даного дослідження, під еволюцією розуміється процес проактивного саморозвитку і самоадаптації інтелектуальних активних компонентів (агентів) відповідно до змін у тому середовищі, де вони мешкають – тобто, у ЄІП.

Модель еволюції [27], що застосована у ЄІПДане середовище підтримує :

- Зміни у обмеженнях на стан агента – *здатності* до виконання роботи
- Зміни у концептуалізаціях (уявленнях) агента про своїх партнерів – учасників суспільства виконавців завдання
- Зміни у *інформаційних ресурсах і їх метаданих*

Еволюція здатностей: Згідно із 10 еволюція здатностей – це процес переходу агента (скажімо, A) із одного стану s_i в інший s_j . Агент *A* як автономний об'єкт, виконує ці переходи згідно із рішеннями, які він сам прийняв під час виконання тієї чи іншої атомарної роботи. Внаслідок цього "манера" виконання агентом A політики f , а також обмеження на політику вхідних параметрів X_f , залежить від стану агента A. Тому, еволюція агента – це еволюція його ролі.

Множина станів агента *A*: $S_A = \{s_1,...,s_n\}$ - це множина триплетів $s_i, i = 1,...,n$:

$$s_i = \{r(X_A), q(F_A), t(F)\}, \tag{2.7}$$

де $r(X_A)$ - множина обмежень, що застосовуються в стані s_i до системних параметрів X_A агента *A* (обмеження на параметри),

$q(F_A)$ - множина обмежень в стані s_i до авторизованих політик агента *A* (обмеження на політики),

$t(F)$ - функція, що визначає переходи зі стану s_i до інших дозволених станів із S_A,

які з'являються після виконання агентом A політик $F = \{f_1,..., f_j,..., f_m\}$.

Еволюція уявлень: Еволюція уявлень близько пов'язана з обстеженням можливостей учасників суспільства по виконанню завдання. Згідно з [10, 11] комунікація між агентами і виконання роботи організована через параметричні відгуки, що містять інформацію про наявні можливості щодо виконання конкретної роботи. Можливість, що повертається виконавцем A користувачу B, це функція від параметрів завдання (політики) $c_A^f = c(X_f)$, $c_A^f \in [0,1]$. Агент (який представляє функціональний компонент реального світу) обстежує можливості своїх співробітників для того щоб назначити роботи виконавцям із вірогідно кращими можливостями у майбутньому. Уявлення про вірогідні можливості агентів-співробітників мають вигляд матриці:

$$\mathbf{C} = \begin{array}{c} \\ A_1 \\ \dots \\ A_i \\ \dots \\ A_n \end{array} \begin{array}{c} \begin{array}{ccccc} w_1 & \dots & w_j & \dots & w_m \end{array} \\ \begin{bmatrix} c_1^1 & & c_1^j & & c_1^m \\ & & \dots & & \\ & \dots & c_i^j = c_{A_i}^{w_j}(X_{w_j}) & \dots & \\ & & \dots & & \\ c_n^1 & & c_n^j & & c_n^m \end{bmatrix} \end{array}, \qquad (2.8)$$

де розмірності n і m зростають в процесі еволюції, відображаючи появлення нових знань про агентів-співробітників (n) і про роботи, які вони, можливо, зможуть виконувати. Верхня границя розмірності n - це кількість агентів-учасників МАС, яку контролює володар матриці **C**. Максимальне значення m - це розмірність множини W дозволених атомарних робіт вищеназваної МАС.

Зміни у інформаційних ресурсах: Зміни в даних і метаданих інформаційних ресурсів підтримуються локально відповідними розподіленими інформаційними системами – постачальниками інформації. В рамках нашого підходу постачальників інформаційних ресурсів представляють агенти – оболонки, які еволюціонують відповідно до змін.

2.5 Проблеми семантичної інтероперабельності

Ще одним із найважлівійших завдань даній роботи є розробка підходів до вирішення проблем семантичної інтероперабельності у визначених суспільствах агентів, що виконують завдання. Цій підрозділ дає систематизований огляд стану проблеми і прецендентів її вирішення, які у майбутньому мають бути застосовани в ЄІП.

Задача спільного використання неоднорідних баз даних, відома сьогодні також як окремий випадок задачі досягнення інтероперабельності інформаційних ресурсів, виникла на початку 80-х р.р. XX сторіччя.

На той час у розвинутих інформаційних системах (ІС) уже була потреба спільного використання декількох (можливо, розподілених) баз даних, керованих різними СКБД, що супроводжувалося значними зусиллями з боку розроблювачів цих інформаційних систем. Прикладами рішення задачі про спільне використання мережних, ієрархічних і реляційних баз

даних, що відносяться до початку 80-х р.р., можуть бути проекти СИЗИФ [49-51], POLYPHEME[29], проект Каліфорнійського університету [33] та інші.

На сьогоднішній день ця задача набула ще більш важливого значення, у зв'язку зі зростанням кількості різноманітних за змістом, структурою, обсягом інформаційних ресурсів (баз даних, баз знань, програмних компонентів і т.д.), створених на різних програмно-апаратних платформах. Інформація, представлена цими ресурсами, багаторазово дублюється, її спільне використання утруднене в силу різних специфікацій інформаційних ресурсів, прийнятих різними розроблювачами. Така ситуація послужила причиною розвитку досліджень в області спільного і повторного використання компонентів інформаційних ресурсів. Важливість даної проблематики була підкреслена в [52] і виділена як один із трьох базових напрямків у дослідженнях на поточне десятиріччя.

Наведемо основні використовувані надалі визначення.

Інтероперабельність означає можливість створення систем з довільних неоднорідних, розподілених компонентів, на основі уніфікованих інтерфейсів [48].

Інтероперабельна інформаційна система (ІС) складається з компонентів, що представляють довільні інформаційні ресурси (програмні компоненти, бази даних, бази знань, файли даних і т.д.), які розглядаються незалежно від апаратно-програмної платформи і розміщення в просторі. Компоненти взаємодіють, обмінюючись заявками.

Задача досягнення інтероперабельності різнорідних інформаційних ресурсів може бути представлена як дві підзадачі:

Досягнення технічної інтероперабельності, тобто забезпечення спільної роботи різнорідних апаратно-програмних платформ. У даному огляді питання технічної інтероперабельності розглядатися не будуть, з огляду на той факт, що саме цій проблемі дотепер була присвячена велика частина зусиль консорціуму Object Management Group, OMG (потрібну інформацію можна почерпнути в [43, 44, 22]).

Досягнення семантичної інтероперабельності, тобто забезпечення спільного використання різнорідних інформаційних ресурсів.

У даній статті пропонується огляд існуючих систем інтеграції неоднорідних баз даних і баз знань з погляду семантичної інтероперабельності і використані в цих системах підходи.

Семантична інтероперабельність: При створенні ІС на неоднорідних інформаційних ресурсах, для досягнення семантичної інтероперабельності необхідно вирішувати проблеми порівняння вмісту цих ресурсів, знаходження відповідностей і вирішення конфліктів між ними, а також проблему сполучення різнорідних ресурсів. Отже, задача побудови семантично інтероперабельної ІС розбивається на 3 частині:

Вироблення специфікацій, в достатній мірі уніфікованих і повних с точки зору конкретної задачі, для всіх інформаційних ресурсів.

Вироблення засобів порівняння можливостей доступних інформаційних ресурсів з потребами прикладних задач, розв'язуваних даною ІС.

Створення несуперечливої й адекватної моделі предметної області задачі шляхом композиції моделей предметних областей, що відповідають конкретним інформаційним ресурсам.

При цьому повна специфікація інформаційного ресурсу буде включати:
- специфікацію його структури і функцій (статичні характеристики);
- специфікацію обмежень цілісності;
- специфікацію поведінки інформаційного ресурсу (динамічні характеристики);
- специфікацію контексту, тобто області, у якій передбачається використання ресурсу.

Сучасні системи інтеграції неоднорідних інформаційних ресурсів використовують концепцію **медіатора** (див. наприклад в [45]), тобто посередника між розподіленими інформаційними ресурсами у межах інтероперабельної системи та користувачами цих ресурсів.

Визначення структури і поведінки конкретного інформаційного ресурсу завжди визначається семантикою предметної області, що її представляє цій ресурс. Специфікація цієї змістовної оболонки даних може бути виконана у вигляді інформаційної схеми (як, наприклад, у структурованих моделях даних типу реляційної). Однак на більш загальному рівні, будь-яка специфікація семантики даних може бути записана на деякій формалізованій мові, наприклад, на численні предикатів 1-го порядку, або на багатосортній логіці. Кожен предикат, що є присутнім у цій специфікації, відповідає деякому семантичному правилу, що визначає несуперечливий стан бази даних.

Як інструмент створення таких узагальнених специфікацій була запропонована ідея використання **онтологічних специфікацій**.

У [48] наводиться визначення **онтологічної специфікації** інформаційного компонента як набору визначень і понять, а також правил (аксіом), зв'язаних з визначеннями і поняттями з предметної області (прикладного контексту).

Термін "онтологія" у даний час використовується в двох контекстах:

У філософському контексті, **онтологія** – система категорій, використовувана для розгляду з урахуванням конкретного бачення світу [37].

У контексті інформаційних систем, **онтологія** – формалізований опис загальноприйнятого розуміння деякої предметної області, за допомогою якого можуть спілкуватися люди, комп'ютерні системи [47]. Програмні компоненти, для взаємодії між собою в рамках інтегрованої системи неоднорідних ресурсів, використовують **онтології**. Передбачається, що онтологія не залежить від мови представлення предметної області.

Онтологія, на відміну від бази знань, не містить ані знань про методи рішення задач, що стосуються предметної області, ані знань, що дозволяють видавати відповіді на прямі запити про предметну область[35].

На відміну від метаданих, таких як тип, розмір атрибута, онтології повинні мати набагато більш багаті засоби вираження семантики даних. Онтологія може бути традиційною ієрархією понять і типів об'єктів, разом з точним описом кожного типу, однак може містити й аксіоми, що задають обмеження на можливі інтерпретації цих понять[35].

За сучасними уявленнями, розрізняють чотири типи онтологій: **онтології предметних областей**, **онтології задач**, **онтології методів рішення** класів задач, і **формальна онтологія** [37].

Онтологія специфічна для домену, або онтологія предметної області (domain-specific ontology) – опис предметної області, що не залежить від її використання.

Онтологія задач (task ontology) – опис термінів предметної області, прийнятої в конкретному класі задач.

Онтологія методів рішення класу задач (problem-solving methods ontology)– опис термінів і правил, у яких задані методи рішення класу задач. При цьому методи рішення задач – незалежні від предметної області специфікації алгоритму рішенні проблеми (задачі), які можна використовувати для різних предметних областей і (можливо) різних класів задач.

Формальна онтологія (formal ontology або top-level ontology) – опис абстрактних понять, таких як "простір", "час", об'єкт", "подія" тощо.

Дослідження в області онтологій: Перші роботи з дослідження онтологій з'явилися на перетину таких галузей науки як штучний інтелект, філософія, логіка і теорія баз даних.

Одним з найперших проектів, у якому використовувалося поняття онтології, є проект CYC [39]. Метою проекту CYC було створення величезної (порядку 100 000 000 аксіом) бази знань про навколишній світ, яку можна було б використовувати в системах штучного інтелекту для того, щоб перебороти обмеженість сприйняття такими системами навколишнього світу в силу відсутності в їхніх базах знань набору загальноприйнятих понять, так званого загальноприйнятого змісту (common sense).

У рамках проекту CYC була розроблена мова представлення знань CycL. База знань була розділена на два рівні: епістемологічний (для визначення понять, їх зв'язків, аксіом, що задають обмеження) і евристичний (представлений набором засобів логічного висновку, таких як "генератор аргументу", "порівняння аргументів", "знаходження протиріч", "відновлення логічних висновків", і функціонального інтерфейсу для спілкування з евристичним рівнем бази знань CYC). База знань використовує онтологію, організовану за принципом колекцій категорій (точніше, натуральних сортів Куайна), упорядкованих за допомогою абстракцій узагальнення / спеціалізації.

Колектив розроблювачів зі Стенфордського університету з кінця 1980 р. займається розробкою і стандартизацією мов представлення знань [25], інструментальними засобами створення і модифікації онтологій [35]. У середовищі Ontolingua [35] можна створювати онтології доменів, класів задач, методів рішення задач.

Особливе місце в дослідженні онтологій займає розробка формальної онтології, "…що розглядається як теорія апріорних форм і суті об'єктів…"[38]. Метою цих досліджень є розробка системи логічних примітивів (предикатів), представлених на деякій формалізованій мові, структурованих на підставі множини розглянутих раніше онтологічних угод про довільну предметну область. Особливу цінність здобуває формалізація визначень тієї чи іншої категорії, для того, щоб при побудові онтології використовувати строгі принципи поділу типів, а не інтуїтивні, евристичні правила [38].

Подальшою задачею є об'єднання цих систем примітивів, і приведення їх до загальної формальної онтології.

Системи інтеграції неоднорідних інформаційних ресурсів: Відповідно до класифікації, запропонованої в [45], існуючі у світі інформаційні системи можна розділити на три покоління, за рівнем інтероперабельності.

1-е покоління: інформаційні системи, засновані на структурованих базах даних. Домінуючий підхід при розподілі даних – використання федеративних систем баз даних. Основа для технічної інтероперабельності – локальні мережі.

2-е покоління: інформаційні системи ґрунтуються на структурованих базах даних, частково структурованих даних (текст, гіпертекст у форматі SGML), на форматах даних, специфічних для конкретної предметної області (наприклад, графіка, відео і т.д.). Дані можуть зберігатися на десятках локальних мереж, об'єднаних між собою. Вважається досягнутою технічна інтероперабельність, основна увага приділяється узгодженню мов звертання до даних і узгодження структур інформаційних компонентів.

3-е покоління: інформаційні системи, засновані на усіх відомих способах комп'ютеризованого збереження даних, особлива увага приділяється підтримці відео / просторових / часових / наукових даних. Дані можуть зберігатися як у глобальних корпоративних мережах, так і в Інтернет. Вважається досягнутою технічна, синтаксична, структурна інтероперабельність, і особлива увага приділяється узгодженню семантики використовуваних компонентів.

Вочевидь, зараз найбільш розвиненими є інформаційні системи 1-го покоління інтероперабельності, але для нас більш цікавими будуть системи 2-го і 3-го поколінь.

Існує декілька проектів систем 2-го покоління, що використовують погляд на світ з боку семантики, закладеної в метаданих, і використовують онтології. До таких проектів відносяться SIMS [30], HERMES [28], InfoSleuth [31], TSIMMIS [34], Information Manifold [40], OBSERVER[42].

Ці проекти надають доступ до гетерогенних і розподілених інформаційних ресурсів.

Розглянемо ті з них, що надають можливість спільного використання неоднорідних баз даних.

SIMS (http://www.isi.edu/sims)

Модель предметної області (application domain) створюється із використанням системи представлення знань для того, щоб забезпечити фіксований словник описів об'єктів предметної області, атрибутів і відносин між об'єктами. Кожному інформаційному ресурсу ставиться у відповідність модель, у якій описана використовувана в цьому ресурсі модель даних, мова запитів, розташування в мережі, приблизні розміри, і т.д., а також зміст полів цього ресурсу в термінах моделі предметної області. Запити в SIMS формулюються загальною мовою високого рівня також в термінах моделі предметної області. SIMS визначає потрібні інформаційні ресурси за допомогою знань, закладених у модель предметної області й у моделях інформаційних ресурсів системи. Потрібні інформаційні ресурси визначаються під час виконання запиту.

HERMES (http://www.cs.umd.edu/projects/hermes)

Зовнішні інформаційні ресурси представлені у вигляді доменів, що виконують визначені функції з визначеними вхідними і вихідними типами даних. Опис зовнішніх ресурсів виконується за допомогою гібридних баз знань [41]. Звертання до цих доменів виконується за допомогою декларативної мови, заснованої на формальній логіці. На технічному рівні інтероперабельності використовується архітектура медіаторів.

INFOSLEUTH (http://www.mcc.com:80/projects/infosleuth)

Інформаційні ресурси доступні на рівні семантичних концепцій, що є досягненням автономії даних. Інформаційні запити формулюються природно, незалежно від структури, розміщення і навіть існування потрібних інформаційних ресурсів. INFOSLEUTH фільтрує ці запити, сформульовані на семантичному рівні, і знаходить потрібні значення в доступних на час запиту інформаційних ресурсах.

У проекті використовується загальна онтологія домену, і локальні відображення схем баз даних у цю загальну онтологію. Система виконує попередню обробку і перетворює дані з окремої бази даних у записи, атрибути яких є концепціями загальної онтології домену.

TSIMMIS (http://www-db.stanford.edu/tsimmis)

Підтримується модель даних і загальна мова запитів для об'єднання інформації зі структурованих і частково структурованих джерел даних. Особлива увага приділяється автоматичному створенню трансляторів і медіаторів для доступу до різнорідних ресурсів. Результуюча інформація представляється в моделі обміну об'єктами (Object Exchange Model).

Information Manifold (http://www.research.att.com/~luy/imhome.html)

Призначена для підтримки інформаційних ресурсів (структурованих і неструктурованих). Архітектура система заснована на базі знань, що містить багату модель предметної області, у термінах якої і відбувається опис ресурсів. Користувач має можливість переглядати інформаційну базу як у виді бази знань, так і у вигляді окремих інформаційних ресурсів, і задавати запит на пошук на декларативній мові запитів. Головною задачею й особливістю цієї системи є можливість оптимізації запиту користувача.

Щодо систем 3-го покоління інтероперабельності відзначимо, що на сьогоднішній день існує декілька, поки тільки рамкових, програм по створенню таких систем. Це:

Knowledge Sharing Effort (http://www-ksl.stanford.edu/knowledge-sharing)

Intelligent Integration of Information (http://mole.dc.isx.com/i3)

Digital Library Initiative (http://www.cise.nsf.gov/iis/dli_home.html)

Як можна побачити, у сучасних системах використовуються дві концепції архітектури медіатора:

Централізований підхід, за яким необхідно існує центральний медіатор. Цей медіатор визначає, до якого ресурсу треба звернутися при відповіді на запит користувача, за допомогою центрального словника даних, або загальної онтології ІС. (TSIMMIS, Information Manifold).

Децентралізований підхід, в якому кожен ресурс має програмного агента, що відображає онтологічну специфікацію ресурсу в загальну онтологію даної ІС. Відповідь на запит формується після комунікації агентів окремих ресурсів із агентом брокера ресурсів, який, на відміну від центрального медіатора, визначає релевантні ресурси в процесі відповіді на запит, виходячи із доступних/релевантних ресурсів (InfoSleuth, SIMS, HERMES).

Системи спільного використання знань: При спільному використанні різних баз даних цінність представляє обсяг і семантика даних, що зберігається в тій чи іншій базі. У той же час, розробка комплексних систем знань пов'язана зі значними зусиллями по формалізації і представленню знань, тому що цінність буде мати якість і повнота наявних концепцій. Тому поряд із системами інтеграції неоднорідних баз *даних*, не менше значення мають дослідження в області спільного використання компонентів баз *знань*.

Проект IBROW3 [32] призначений для розробки інтелектуального сервісу, що дозволить використання компонентів знань від різних розроблювачів, з використанням технологій WWW. У рамках проекту використовуються і онтології предметних областей, і онтології класів задач, і онтології методів рішення класів задач. Як базову мову для опису бібліотеки методів рішення задач в цьому проекті використовують мову UPML.

Паралельно IBROW3 розвивається проект DARPA за назвою High-Performance Knowledge Bases (HPKB) (http://www.teknowledge.com:80/HPKB). Метою проекту є масштабування повторно використовуваних компонентів знань і доступ до них за допомогою Internet.

Доступ до різнорідних даних в Інтернет: Особливе місце в системах інтеграції неоднорідних компонентів відіграває Інтернет і World – Wide Web. Величезна кількість інформації розсіяна по Мережі, і тут важливість має пошук релевантної інформації і відсікання непотрібної інформації. Сучасні машини пошуку (search engines), застосовувані на пошукових серверах, мають потребу в додаткових засобах пошуку тільки релевантної інформації. Онтології предметних областей, як засоби специфікації онтологічних угод між постачальниками інформації і користувачами, можуть змінити ситуацію до кращого. Використання онтологій дозволяє користувачу сформулювати свій запит на більш високому рівні абстракції, ніж це можливо при пошуку по ключових словах.

Розглянемо приклади систем, що використовують онтології для роботи з Інтернет.

OBSERVER (http://siul02.si.ehu.es/~jirgbdat/OBSERVER)

Ця система пропонує підхід використання безлічі вже існуючих онтологій для доступу до гетерогенних, розподілених і незалежно розроблювальних репозиторіях даних [42]. Реалізація такого підходу - ідеологія брокера онтологій предметних областей. Передбачається, що існує безліч заздалегідь створених онтологій предметних областей, і користувачу необов'язково "підбудовуватися" під конкретну онтологію. Користувач формулює свій запит на деякій мові, у термінах однієї чи декількох онтологій, і брокер «шукає» релевантні інформаційні ресурси, виконуючи транслювання запиту в придатні онтології, а в разі потреби, і сполучення декількох онтологій для більш точної відповіді на запит.

OntoSeek [36]

Ця система розроблена для контекстного отримання інформації з он-лайнових “жовтих сторінок” та каталогів продуктів. Система може працювати як з однорідними, так і з неоднорідними каталогами продуктів. Для точної фіксації контексту може бути застосований інтерактивний підхід, коли користувач поступово уточнює зміст ключових слів, за допомогою лінгвістичної бази даних WordNet (http://www.let.uva.nl/~ewn). WordNet – це лінгвістична база даних, що складається із сінсетів(synsets) – груп слів, еквівалентних за змістом. WordNet є водночас і лексичним словником (створеним для декількох європейських мов), і онтологією, що представляє зв’язки між словами у словнику. Опис ресурсу реалізується у вигляді лексичного концептуального графа[46], де вершини відповідають словам, а іменовані дуги – семантичним відносинам між словами (наприклад, відносини типа “частина”, або “підклас”, або ін.), назви вершин і дуг також беруть із WordNet, під час створення концептуального графа конкретного ресурсу. Знаходження ресурсів, відповідних до запиту користувача, базується на порівнянні онтологій (лексичних концептуальних графів) цих ресурсів. А саме, при відборі ресурсів, відповідних до запиту користувача, OntoSeek виконує порівняння концептуального графа запиту із існуючими концептуальними графами ресурсів або з частинами цих графів.

OntoSeek має централізований сервер, на якому знаходиться база даних лексичних концептуальних графів відомих системі ресурсів, але створення таких графів виконується з боку клієнта.

Підхід, використаний в OntoSeek, відрізняється від підходу, який застосовується у моделі W3C Resource Description Framework (W3C RDF, http://www.w3c.org). У RDF опис структури даних (тобто, схема даних у вигляді <subject, predicate, object>), додається прямо у HTML/XML документ, а не зберігається окремо. Ніяких додаткових умов щодо семантичної узгодженості даних RDF не вимагає.

Підсумки: У даному короткому огляді була зроблена спроба конкретизації стану сучасних систем інтеграції неоднорідних баз даних і баз знань, і підходів, використаних у цих системах.

На сьогодні, задача інтеграції неоднорідних ресурсів має багато рішень. Однак, серед узагальнюючих факторів при рішенні даної задачі виділимо такі:

Перший фактор – використання онтологій як специфікацій предметних областей, задач, методів рішення задач.

Другий фактор – розвиток інтернет-технологій, що дозволяють зробити різнорідні інформаційні ресурси «ближче до користувача».

Третій фактор – застосування систем інтелектуальних агентів для реалізації архітектури медіатора неоднорідних розподілених інформаційних ресурсів.

Втім, дійсний момент – тільки перший етап у використанні цих факторів повною мірою.

2.6 Висновки

Як відзначалося вище, ключовими проблемами в організації взаємодії агентів, як автономних сущностей, є реалізація засобів передачи впливів і сообщеній (комунікації), а також узгодження спільних робіт (координації) між агентами що діють спільно для вирішення спільної проблеми, або виконання спільного завдання.

Розглянуті принципи і методи організації взаємодії агентів, дозволили розробити формальні моделі комуникации, координації та еволюції об'єктів функціональної системи. Розглянути підходи до вирішення проблем семантичної інтероперабельності дозволили сформулювати вимоги до контекстних специфікацій знань у вигляді онтологій і методів вирішення проблем.

3 Приклади моделювання процесів у ЄІП

В даному підрозділі розглянути приклади моделювання типових процесів, які мають місце у ВУЗі. Перший приклад - є приклад моделювання процесу планування науково - технічної роботи [25], другій - є приклад, пов'язаний із процесами дистанційного навчання у віртуальному університеті [27].

3.1 Моделювання процесу планування проекту

Одне із застосувань даного середовища моделювання - планування бізнес-процесів. Головна ідея близька до ідеї, використаної у проекті ADEPT [53]: моделювати бізнес-процеси як колекції автономних агентів, що вирішують проблеми, і здатних спілкуватися для моделювання виконання бізнес-процесу. Відповідний план бізнес-процесу може бути автоматично створено потім.

Розглянемо, як наше середовище можна використати для планування процесу розробки інформаційної системи. Будемо вважати, що організаційна структура, яка може виконати цей проект є така, як показано на Рис. 3.1. - Відділ ІС. Зробимо також такі припущення:

Відділ ІС, поданий на Рис. 3.1. є одним із кандидатів на виконання цього проекту. З точку зору змагання, відділ ІС, а також інші учасники вважаються агентами - учасниками, що представляють функціональні компоненти суспільства більш високого рівня.

Суспільство агентів відділу ІС складається з таких функціональних компонентів: менеджер

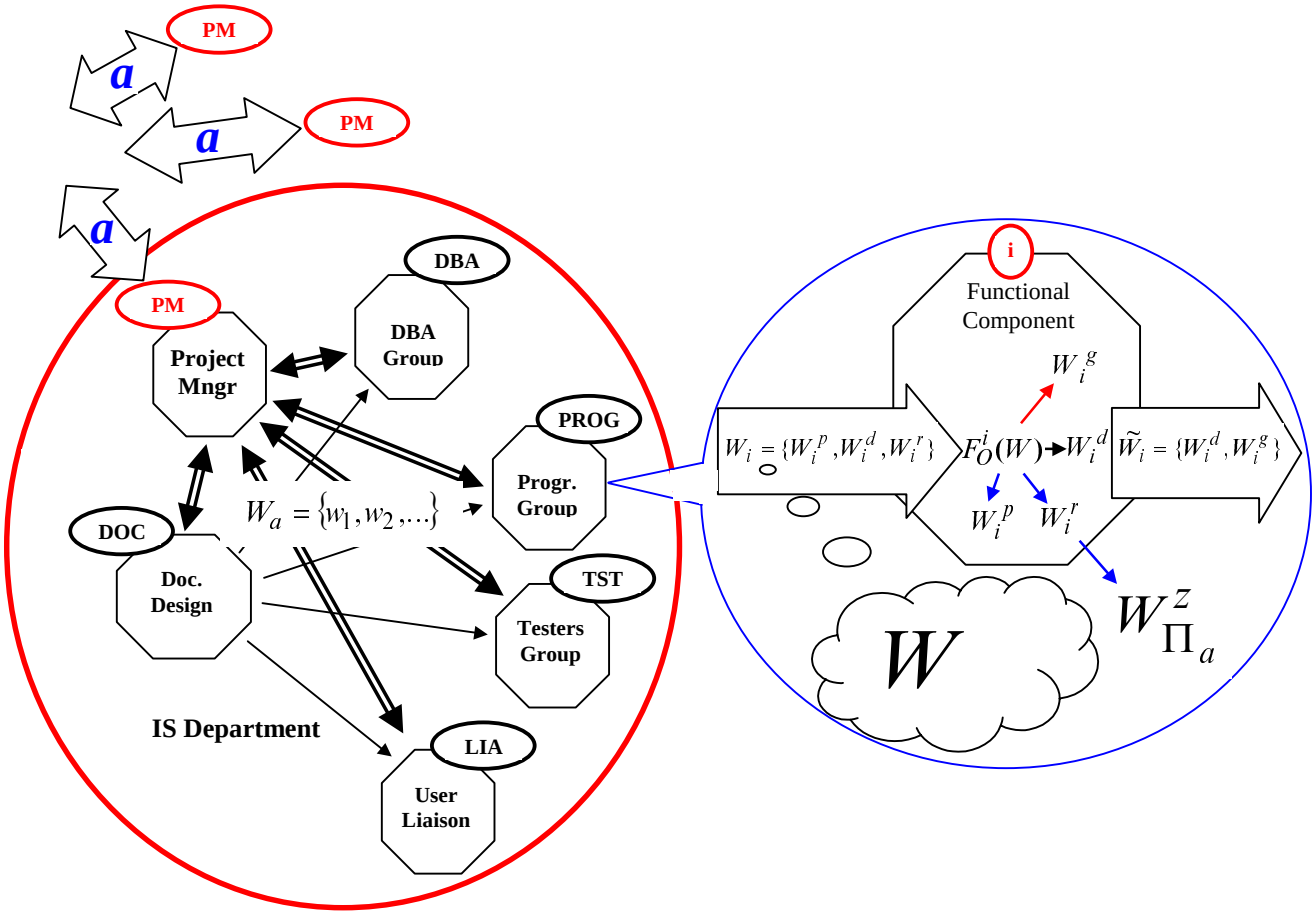

Рисунок 3.1 Модель компоненти/функціональної системи згідно [25].

проекту (агент **PM**), група адміністратора баз даних (агент **DBA**), група програмістів (агент **PROG**), група створення документації (агент **DOC**), група тестерів (агент **TST**), група зв'язку із користувачами (агент **LIA**)

Якщо на деякому етапі виникає потреба розглянути один із функціональних компонентів детальніше, цей компонент можна промоделювати як функціональну систему – у вигляді суспільства агентів.

Процес починається в момент часу t_0 із зовнішнього впливу $W_a = \{w_0 = '\mathit{Propose\_IS\_Development\_Plan}', X_0, Y_0\}$

З такими параметрами і описом результатів

$X_0 = \{\textbf{\textit{budget}} =< \mathit{figure} >, \textbf{\textit{duration}} =< \mathit{figure} >, \textbf{\textit{Proposal\_Template}} =< \mathit{file\_name} >,$

$\textbf{\textit{Proposal\_Descr}} =< \mathit{file\_name} >\}$;

$Y_0 = \{\mathit{Possibility}(\textbf{\textit{budget}}, \textbf{\textit{duration}}), \mathit{Proposal}(\textbf{\textit{Proposal\_Template}})\}$

що надходить до сенсорного входу агенту **PM. PM** виконує як найменш такі дії:

1. **Перевірка вхідного впливу.** На цьому етапі **завдання PM** - перевірити, чи узгоджується вхідний вплив, його параметри і опис результатів із роллю **PM** і його поточним станом. Далі поведінка **PM** (тобто, його макромодельна програма), може бути одна з двох: вхідний вплив приймається, або відхиляється. Не будемо розглядати ситуацію, коли зовнішній вплив відхиляється. Будемо надалі вважати, що всі вхідні впливи приймаються і виконуються відповідні макромодельні програми.

2. **Виконання роботи і зміна стану агента.** На цьому етапі макромодель **PM** $F_O^{PM}(W_a)$ (відповідно із онтологією планування програмного проекту, доступ до якою забезпечує агент онтології) розкладає вхідну роботу, тобто генерує множину робіт, що відповідають етапам проекту, згідно із правилом (1.1а), де:

$W_{PM}^p = \{w_0 = '\mathit{Propose\_IS\_Development\_Plan}', X_0, Y_0\}$, $W_{PM}^d = \varnothing$, $W_{PM}^r = \varnothing$,

$W_{PM}^g =$ $\{ w_1 = (\mathit{'Assemble\ Project\ Proposal'}, X_1, Y_1),$
$w_2 = (\mathit{'Choose\ best\ DB\ Schemata\ Plan\ Bid'}, X_2, Y_2),$
$w_3 = (\mathit{'Choose\ best\ Software\ Model\ Plan\ Bid'}, X_3, Y_3),$
$w_4 = (\mathit{'Choose\ best\ REQ\ Analyses\ Plan\ Bid'}, X_4, Y_4),$
$w_5 = (\mathit{'Analyse\ requirements'}, X_5, Y_5),$
$w_6 = (\mathit{'Design\ database\ schemata'}, X_6, Y_6),$
$w_7 = (\mathit{'Design\ software\ model'}, X_7, Y_7),$
$w_8 = (\mathit{'Program\ the\ software'}, X_8, Y_8),$
.....
$w_{15} = (\mathit{'Perform\ customers\ training\ '}, X_{15}, Y_{15})\}.$

На цьому етапі виконання макромодельної програми зміна стану агента **PM** від s_0 до s_1 шляхом додання додаткових обмежень може бути зроблена, якщо це відповідає правилу поведінки, яке записано у відповідній онтології, але вбудоване в макромодельну програму.

3. Генерація компонентів матриці $\mathbf{K}_{PM}(t_0)$. На цьому етапі макромодель **PM** генерує $\mathbf{K}_{PM}(t_0)$, як показано на Рис. 3.5.

$$\mathbf{K}_{PM} = \begin{array}{c} \\ \textbf{DBA} \\ \textbf{PROG} \\ \textbf{PM} \\ \textbf{DOC} \\ \textbf{TST} \\ \textbf{LIA} \end{array} \begin{array}{c} \begin{array}{cccccccccc} w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & w_7 & w_8 & \dots & w_{15} \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & & 1 \end{bmatrix} \end{array}$$

Рисунок 3.2 Наміри і можливості **PM** до завдання $W_a = \{'developIS', X, Y\}$ в $]t_0, t_0 + \Delta t]$

$$\Omega_a = \begin{array}{c} \\ \Theta_{DBA} \\ \Theta_{PROG} \\ \Theta_{PM} \\ \Theta_{DOC} \\ \Theta_{TST} \\ \Theta_{LIA} \end{array} \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & & 1 \end{bmatrix}$$

Рисунок 3.3 Стан системи (відносно до W_a) в момент часу $t_0 + \Delta t$.

Матрицю $\mathbf{K}_{PM}(t_0)$ можна інтерпретувати так: на наступному етапі моделювання , в момент часу $t_0 + \Delta t$ роботи w_1, w_2, w_3, w_4 виконуються; роботи w_5, w_6, w_7 перенаправляються декільком функціональним компонентам: w_5 - агентам **DBA** і **LIA**, w_6 - агентам **DBA**, **PROG** і **LIA**, w_7 - агентам **DBA**, **PROG** і **LIA**. Ціллю такого перенаправлення може бути наприклад те, що агент **PM** точно не знає, у чому полягаю обов'язки **DBA**, **PROG**, **LIA**, і хоче отримати деякий досвід у цьому для використання у майбутньому. Однак, ми не вважаємо, що агент **PM** може перенаправити роботи декільком виконавцям паралельно, щоб кожний із виконавців виконав частину роботи, тому що це суперечить принципу атомарності роботи. Роботи w_5, w_{15} були назначені явно агентам **PROG** і **LIA**.

Матриця станів системи $\Omega_a(t_0 + \Delta t)$, пов'язана із процесом $W_a = \{'Propose\_IS\_Development\_Plan', X, Y\}$ буде виглядати як на Рис.3.3., оскільки всі агенти, крім **PM** відбувають у «ледачому» стані у момент часу $t_0 + \Delta t$.

Розглянемо діяльності агентів **PM, DBA, LIA** на етапі $t_1 = t_0 + \Delta t$.

PM приймає множину робіт:

$W_{PM} = \{ w_1 = ('Assemble\ project\ proposal', X_1, Y_1),$
$w_2 = ('Choose\ best\ DB\ Schemata\ Plan\ Bid', X_2, Y_2),\},$
$w_3 = ('Choose\ best\ Software\ Model\ Plan\ Bid', X_3, Y_3),$
$w_4 = ('Choose\ best\ REQ\ Analyses\ Plan\ Bid', X_4, Y_4)\}$

із параметрами і описом результатів для роботи w_1:

$X_1 = \{\textbf{budget} =< figure >, \textbf{duration} =< figure >,$
$\textbf{Proposal\_Template} =< file\_name >,$
$\textbf{Proposal\_Descr} =< file\_name >,$
$\tilde{Y}_2, \tilde{Y}_3, \tilde{Y}_4, \tilde{Y}_8, ..., \tilde{Y}_{15}\},$

$Y_1 = \{Possibility(\textbf{budget}, \textbf{duration}), Proposal(\textbf{Proposal\_Template})\}$

Оскільки в момент t_1 параметри $\tilde{Y}_2, \tilde{Y}_3, \tilde{Y}_4, \tilde{Y}_8, ..., \tilde{Y}_{15}$ ще не містять коректних значень, скінчено-станова машина F_X агента **PM** переходить у стан відхилення і класифікує роботу w_1 як приналежну до множини робіт W_{PM}^d, що перенаправляються. Таким чином, робота w_1 реально відкладається як мінімум на період Δt макромодель **PM** додає 1 до елементу d_1 вектора затримок

процесу D_a. Робота w_1 знову назначається агентові **PM** для виконання на наступному етапі моделювання t_2. Це саме повторюється для робіт w_2, w_3, w_4.

DBA приймає набір робіт $W_{DBA} = \{ w_5 = (\textit{'Analyse requirements'}, X_5, Y_5),\ w_6 = (\textit{'Design database schemata'}, X_6, Y_6),\ w_7 = (\textit{'Design software model'}, X_7, Y_7)\}$з відповідними параметрами і описом результатів:

$X_5 = \{\ x_5^1 = (\textbf{\textit{budget}} = <b>),\ x_5^2 = (\textbf{\textit{duration}} = <d>),$

$x_5^3 = (\textbf{\textit{Plan\_Template}} = <file\_name>),\ x_5^4 = (\textbf{\textit{Step\_Descr}} = <file\_name>)\ \},$

$Y_5 = \{\ y_5^1 = (\textit{Possibility}(\textbf{\textit{budget}}, \textbf{\textit{duration}})),\ y_5^2 = (\textit{Plan\_File\_Name}(\textbf{\textit{Plan\_Template}}))\ \};$

$X_6 = \{\ x_6^1 = (\textbf{\textit{budget}} = <b>),\ x_6^2 = (\textbf{\textit{duration}} = <d>),$

$x_6^3 = (\textbf{\textit{Plan\_Template}} = <file\_name>),\ x_6^4 = (\textbf{\textit{Step\_Descr}} = <file\_name>)\ \},$

$Y_6 = \{\ y_6^1 = (\textit{Possibility}(\textbf{\textit{budget}}, \textbf{\textit{duration}})),$

$y_6^2 = (\textit{Plan\_File\_Name}(\textbf{\textit{Plan\_Template}}))\ \};$

$X_7 = \{\ x_7^1 = (\textbf{\textit{budget}} = <b>),\ x_7^2 = (\textbf{\textit{duration}} = <d>),$

$x_7^3 = (\textbf{\textit{Plan\_Template}} = <file\_name>),$

$x_7^4 = (\textbf{\textit{Step\_Descr}} = <file\_name>)\ \},$

$Y_7 = \{\ y_7^1 = (\textit{Possibility}(\textbf{\textit{budget}}, \textbf{\textit{duration}})),$

$y_7^2 = (\textit{Plan\_File\_Name}(\textbf{\textit{Plan\_Template}}))\ \};$

	0.5*d*	0.8*d*	1.0*d*	1.5*d*	1.8*d*
0.5*b*	*p*	*p*	*p*	*p*	*p*
0.8*b*	*p*	*p*	*p*	*p*	*p*
1.0*b*	*p*	*p*	*p*	*p*	*p*
1.5*b*	*p*	*p*	*p*	*p*	*p*
1.8*b*	*p*	*p*	*p*	*p*	*p*

Possibility: значення $p \in [0,1]$

Рисунок 3.4 Приклад опису результатів.

де: *Possibility*(***budget***, ***duration***) - опис результата, зображений на Рис. 3.4.; *Plan_File_Name* (***Plan_Template***) може бути або стрічкою «*DUMMY* », якщо план не було розроблено, або може бути стрічкою, що містить коректне ім'я файла, створене згідно із ***Plan_Template***.

Обробка політики *'Analyse requirements'* скінченно-становою машиною F_A агенту **DBA**, веде до відхилення цієї політики, оскільки в нашому прикладі було прийнято, що ця політика не виконується агентом **DBA**. Виконавець політики відхилення генерує вектор результатів $\tilde{Y}_5 = \{\mathbf{O}, "DUMMY"\}$. Та же сама реакція буде проведена при обробці роботи *'Design software model'*: $\tilde{Y}_7 = \{\mathbf{O}, "DUMMY"\}$, де O - матриця, створена згідно правилу, що дано на Рис. 3.4., і містить нульові значення можливостей: $\mathbf{O}_{ij} = 0, i = 1,..,5,\ j = 1,...,5$. Робота *'Design database schemata'* виконується агентом **DBA**, і отриманий такий результат:

$$\tilde{Y}_6 = \{ \begin{bmatrix} 0.1 & 0.1 & 0.4 & 0.6 & 0.35 \\ 0.3 & 0.35 & 0.4 & 0.8 & 0.35 \\ 0.3 & 1.0 & 1.0 & 0.8 & 0.35 \\ 0.35 & 1.0 & 1.0 & 0.85 & 0.35 \\ 0.4 & 1.0 & 1.0 & 0.9 & 0.35 \end{bmatrix}, "DBSCH\_PLAN.XLS"\} \quad , \qquad (3.1)$$

Агент **LIA** приймає такий самий як у **DBA** набір робот $W_{LIA} = W_{LIA} = \{ w_5 = (\textit{'Analyse requirements'}, X_5, Y_5),\ w_6 = (\textit{'Design database schemata'}, X_6, Y_6),\ w_7 = (\textit{'Design software model'}, X_7, Y_7)\}$. Виконується агентом LIA, і отриманий такий результат:

$$\Omega_a = \begin{matrix} \Theta_{DBA} \\ \Theta_{PROG} \\ \Theta_{PM} \\ \Theta_{DOC} \\ \Theta_{TST} \\ \Theta_{LIA} \end{matrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 \end{bmatrix} \qquad W^{z}_{\Pi_a} = \begin{matrix} DBA \\ PROG \\ PM \\ DOC \\ TST \\ LIA \end{matrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & & 0 \end{bmatrix}$$

Рисунок 3.5. Стан системи і невиконані роботи (відносно до W_a) в $t_1 + \Delta t$.

$$\Omega_a = \begin{matrix} \Theta_{DBA} \\ \Theta_{PROG} \\ \Theta_{PM} \\ \Theta_{DOC} \\ \Theta_{TST} \\ \Theta_{LIA} \end{matrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & & -1 \end{bmatrix}$$

Рисунок 3.6 Стан системи (відносно до W_a) в $t_2 + \Delta t$.

$$\Omega_a = \begin{matrix} \Theta_{DBA} \\ \Theta_{PROG} \\ \Theta_{PM} \\ \Theta_{DOC} \\ \Theta_{TST} \\ \Theta_{LIA} \end{matrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & & 0 \\ 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 \\ 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & & -2 \end{bmatrix}$$

Рисунок 3.7 Стан системи (відносно до W_a) в $t_3 + \Delta t$.

$$\tilde{Y}_5 = \{ \begin{bmatrix} 0.1 & 0.15 & 0.4 & 0.6 & 0.55 \\ 0.1 & 0.55 & 0.4 & 1.0 & 0.65 \\ 0.2 & 0.9 & 1.0 & 1.0 & 0.70 \\ 0.35 & 1.0 & 1.0 & 1.0 & 0.75 \\ 0.4 & 1.0 & 1.0 & 1.0 & 0.75 \end{bmatrix}, "REQ\_PLAN.XLS" \} \quad , \tag{3.2}$$

а роботи w_3, w_4 відхиляються як неналежні з результатами $\tilde{Y}_3 = \{\mathbf{O}, "DUMMY"\}$, $\tilde{Y}_4 = \{\mathbf{O}, "DUMMY"\}$. Матриця станів системи $\Omega_a(t_1 + \Delta t)$ і матриця відхилених робіт $W^{z}_{\Pi_a}(t_1 + \Delta t)$ мають вигляд такий, як на Рис. 3.5.

В момент часу t_2 агент **PM** отримує завдання:

$W_{PM} = \{ w_1 = (\textit{'Assemble project proposal'}, X_1, Y_1),$

$w_2 = (\textit{'Choose best DB Schemata Plan Bid'}, X_2, Y_2),\},$

$w_3 = (\textit{'Choose best Software Model Plan Bid'}, X_3, Y_3),$

$w_4 = (\textit{'Choose best REQ Analyses Plan Bid'}, X_4, Y_4)\}$

з

$X_2 = \{\tilde{Y}_5^{DBA}, \tilde{Y}_5^{LIA}\},$

$Y_2 = \{ y_2^1 = (\textit{Possibility}(\textbf{\textit{budget}}, \textbf{\textit{duration}})),\ y_2^2 = (\textit{Plan\_File\_Name}(\textbf{\textit{Plan\_Template}}))\};$

$X_3 = \{\tilde{Y}_6^{DBA}, \tilde{Y}_6^{PROG}, \tilde{Y}_6^{LIA}\},$

$Y_3 = \{ y_3^1 = (\textit{Possibility}(\textbf{\textit{budget}}, \textbf{\textit{duration}})),$

$y_3^2 = (\textit{Plan\_File\_Name}(\textbf{\textit{Plan\_Template}}))\};$

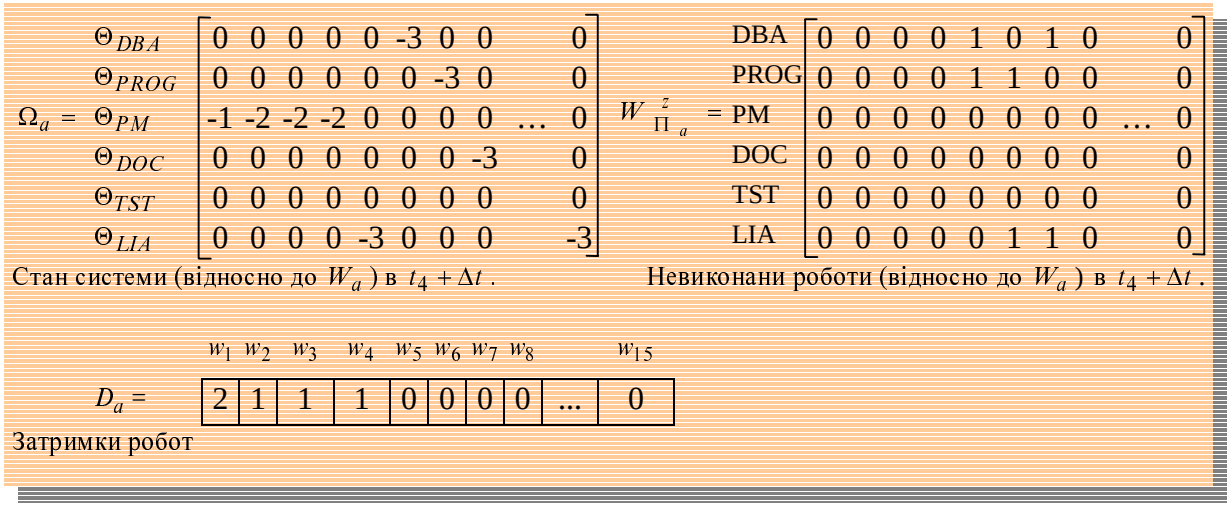

Рисунок 3.8 Стан системи, невиконані роботи і затримки робот (відносно до W_a) після того, як W_a виконана.

$$X_4 = \{\tilde{Y}_7^{DBA}, \tilde{Y}_7^{PROG}, \tilde{Y}_7^{LIA}\},$$

$$Y_4 = \{ y_4^1 = (Possibility(\textbf{\textit{budget}}, \textbf{\textit{duration}})),$$
$$y_4^2 = (Plan\_File\_Name(\textbf{\textit{Plan\_Template}}))\};$$

і в змозі виконати роботи w_2, w_3, w_4. Робота w_1 знову затримана і запланована на наступний момент моделювання. Тепер завдання для макромоделей агента **PM** w_2, w_3, w_4 - вибрати оптимальні плани із прийнятих заявок, згідно із запланованими відгуками *Possibility*. В нашому випадку завдання спрощене тим, що кожен агент виконує тільки одну роботу: LIA виконує w_2, **DBA** виконує w_3, і **PROG** виконує w_4. Результати цього етапу подані на Рис. 3.6. Зверніть увагу на те, що функціональні компоненті **DBA**, **PROG**, **LIA** є «ледачими» в інтервал часу $]t_2, t_2 + \Delta t]$, хоча вони здатні виконати ще деякі роботи.

Подібна оптимізація роботи w_1 виконується макромоделлю агента **PM** в інтервал часу $]t_3, t_3 + \Delta t]$. Ця робота полягає в тому, щоб об'єднати в один план всі плани, запропоновані учасниками суспільства $\tilde{Y}_2, \tilde{Y}_3, \tilde{Y}_4, \tilde{Y}_8, ..., \tilde{Y}_{15}$. Стан системи представлено на Рис. 3.7. Результат – значення *Possibility* у найгіршому випадку буде таке:

$$\tilde{y}_1^1 = \min(\tilde{y}_2^1, \tilde{y}_3^1, \tilde{y}_4^1, \tilde{y}_8^1, ..., \tilde{y}_{15}^1) = \begin{bmatrix} 0.1 & 0.15 & 0.4 & 0.6 & 0.3 \\ 0.1 & 0.5 & 0.4 & 0.75 & 0.35 \\ 0.2 & 0.75 & \vdots & 0.8 & 0.35 \\ 0.3 & 0.9 & 1.0 & 0.8 & 0.35 \\ 0.35 & \cdots & 1.0 & 0.85 & 0.35 \end{bmatrix} \quad (3.3)$$

Хоча завдання W_a в момент часу t_4 вже було виконане, потрібен ще один етап моделювання, щоб перевірити, чи всі агенти припинили абсорбувати вхідні роботи. Результати фінального етапу показані на Рис. 3.8.

Аналізуючи результати моделювання процесу планування, помітимо таке:

1. Параметричні відгуки для моделювання координації і торгівлі. Результуюче значення *Possibility* (3.3) свідчить про те, що, пропонований програмний проект може бути безумовно виконано з підвищеним бюджетом (1.8**b**) навіть швидше, ніж треба - 0.8d. У випадку, якщо бюджет проекту підвищувати не можна, проект може бути виконано за потрібний час, з бюджетом, який починається з 1.0**b**. Взагалі кажучи, параметричні відгуки, які забезпечуються нашим середовищем, роблять дуже простими моделювання протоколів координації (такої, як CNP або брокерство [19]), або аукціону через потоки робіт.

Task/Work	t_1	t_2	t_3	t_4
Tasks Specification	PM			
REQ Analyses Planning		LIA		
DB Schemata Design Planning		DBA		
Software Model Design Planning		PROG		
Programming Planning		PROG		
.....		...		
Customers' Training Planning		LIA		
REQ Analyses Plan Optimisation			PM	
Software Modelling Plan Optimisation			PM	
DB Schemata Design Plan Optimisation			PM	
Project Plan Assembly				PM

Рисунок 3.9 Автоматично згенерований план процеса приготування пропозиції щодо проекту.

2. Навчання і еволюція поведінки. Аналіз $W^{z}_{\Pi_a}$ показав, що роботи w_5, w_6, w_7 були відхилені деякими агентами, хоча врешті-решт, всі роботи були виконані. Слідство таке: $W^{z}_{\Pi_a}$ може бути використано як приклад для більш акуратного призначення робіт в моделюванні процесу планування – агент **PM** вже вивчив можливості учасників суспільства, пов'язаних із виконанням загального завдання і адаптував свою поведінку. Тому ми будемо вважати роботу w_i такою, що не виконується суспільством агентів, пов'язаних до процесу Π_a тільки тоді, коли стовпчик i у $W^{z}_{\Pi_a}$ не містить нульових значень. Однак, невідомо нічого про поведінку суспільства, якщо нові агенти – учасники з новими здатностями (наприклад, що частково перекривають роботи, розміщені в $W^{z}_{\Pi_a}$) приєднуються до команди. Поки що ми вважаємо, що з моменту початку процесу і до моменту його виконання контингент функціональних компонентів не змінюється.

3. Оцінка завантаженості агентів. Оцінка завантаженості агентів, пов'язаних із процесом Π_a - згідно з діаграмою, представленою на Рис. 3.10. – показує, що середня завантаженість агентів - 37.5 %.

4. Планування методом симуляції. Один із додаткових результатів, який може бути отримано при моделюванні процесу планування за допомогою симуляції, є чорновий план створення проекту (див. Рис. 3.9.).

5. Акти комунікації як послідовності робіт. Акти комунікації між агентами, що виконують процес, моделюються як послідовності конкретних робіт, що виконуються у різні моменти часу. В нашому прикладі, виконання роботи w_3

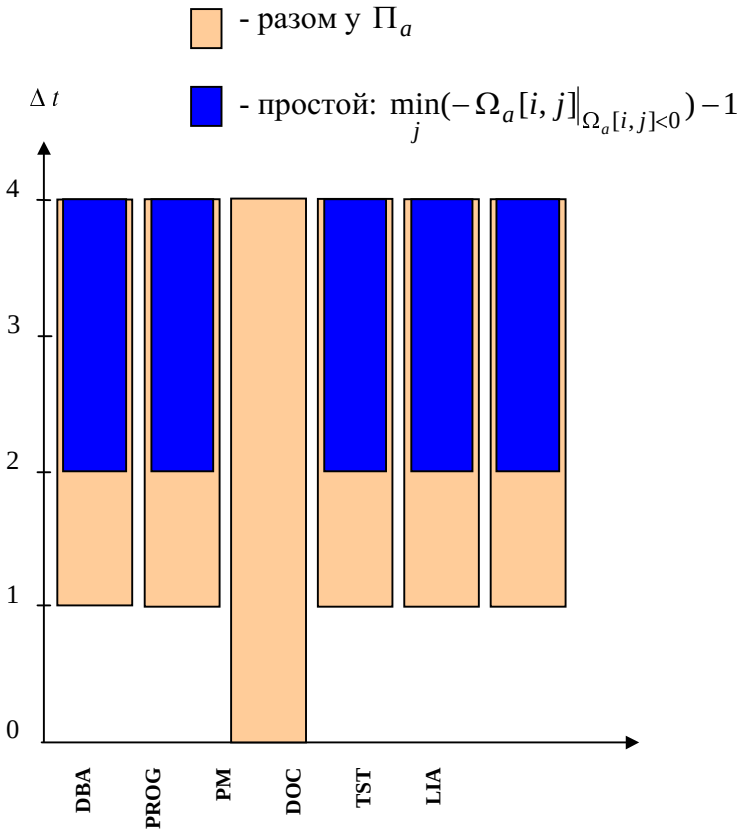

Рисунок 3.10 Навантаження агентів в Π_a.

було віднесено до (див. Секцію 1) ***недетермінованих запитів із аналізом результатів*** w_6 до агентів **DBA**, **PROG** і **LIA** (Рис. 14a), де аналіз результатів виконує агент **PM**. Наш приклад показав, що доцільно виконувати роботи w_3 і w_6 у зворотному порядку, щоб зняти необхідність синхронізації вхідного впливу із відповідною реакцією. Як видно із Рис.14b, ці роботи і були виконані у «інверсному порядку». Також слід відмітити механізм встановлення необхідних затримок для елементів, що знаходяться у «стеці» робіт. В нашому випадку, середовище використовує процедуру F_X верифікації вхідних параметрів для затримки виконання роботи w_3. Ця робота не буде виконана, доки в $\tilde{Y}_6^{DBA}, \tilde{Y}_6^{PROG}, \tilde{Y}_6^{LIA}$ не будуть розміщені коректні значення, тобто доки агенти **DBA**, **PROG** і **LIA** не завершать виконання роботи w_6, яку отримали від агента **PM** в момент часу $t_1 + \Delta t$ (див. Рис. 3.11b). У випадку, якщо роботи не мають параметрів, наше середовище забезпечує альтернативний механізм для створення послідовності - обмеження на стани агентів (див. Рис. 3.11). Процес зміни стану тоді – це використання деяких обмежень, що мають скінченний період життя, і мають такий вигляд:

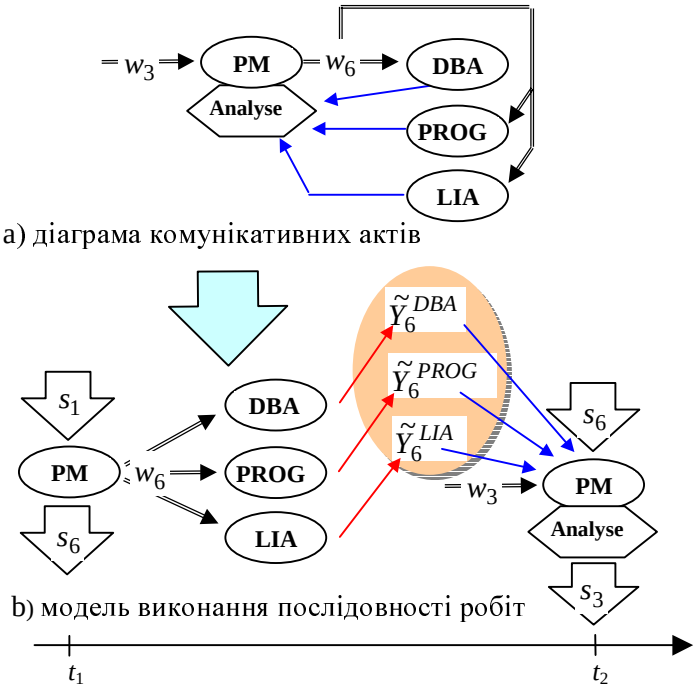

Рисунок 3.11 Моделювання недетермінованого запиту із аналізом результатів

$$s_6 : \left\{ \begin{array}{l} \ldots \\ \texttt{AddConstraint} = \texttt{'Reject } w_3\texttt{'}, \texttt{LifeTime} = 1, \\ \ldots \end{array} \right. \qquad (3.4)$$

якщо викликання виконавців роботи w_6 було комунікативним актом типу ***директива***.

6. Узгодженість із FIPA. Модель послідовності робіт, зображена на Рис. 3.11., за нотацією подібна до Протоколу Мережі Контрактів FIPA [19]. Припущення були такі, що, по-перше, ми можемо моделювати комунікативні акти FIPA ACL за допомогою нашого підходу, а по-друге – ми можемо використати ACL як транспортну мову для комунікацій між агентами, в рамках суспільств, що будуть моделюватися в нашому середовищі.

7. Підготовка роботи «ручним способом». Навіть із розглянутого простого прикладу можна помітити, що на практиці треба виконати багато «ручної» роботи, до того, як функціональна системи буде готова до роботи. Макромоделі, онтології, обмеження на стан, описи параметрів, результатів, ролей повинні бути приготовлені заздалегідь і збережені у базі знань агента / суспільства. Щоправда, більшу частина такої роботи треба виконати один раз, і результат може потім бути використаний іншими агентами /суспільствами (для різних завдань). Треба також помітити, що не всі атомарні роботи можна виконати за допомогою макромодельної програми. Агентів, що використовуються як персональні асистенти, забезпечують помітну допомогу у виконанні рутинної частини роботи. Наприклад, виконання роботи w_6 вимагає участі людини - спеціаліста, для виконання неформальної частини роботи по підготовці плану.

3.2 Моделювання процесу конкурсного відбору аспірантів

Для аналізу припустимості використання ЕІП к імплементації процедур дистанційного навчання і створення Віртуального Університету розглянемо одну із типових функцій -- процес конкурсного відбору аспірантів [27]. Головною причиною вибору саме цього прикладу було прийняття того факту, що ВУ повинен саморегулюватися для того, щоб бути успішним. Процеси управління ВУ потребують аналізу відгуків від процесів дистанційного навчання та викладання, для того, щоб виконати вимоги студентів. З другого боку, процедури доставки курсів і інших відомостей студентам, потребують зворотного зв'язку із блоками керування ВУ. Набір аспірантів можна розглянути як управлінську процедуру (наприклад, як набір співробітників). Нижче буде показано, що цей процес забезпечить нові відомості про курсі, які необхідно ввести,

Будемо вважати, що пошукачі (ті, що бажають стати аспірантами) можуть працювати з ЕІП, контактуючи із факультетами, які вони обирають, через агентів – посередників, і виражати свою потребу бути аспірантом.

Віртуальна кафедра: Заздалегідь зазначимо, що Віртуальна кафедра - це МАС, що складається принаймні з таких акторів (Рис. 3.12):

- Секретар – Агент-Посередник (**РА**)
- Професори (**PRA**), Асистенти (**АА**), розроблювачі курсів (**СМА**), Бібліотекар (**LA**) – агенти середнього рівня

МАС факультету також містить сервісні блоки, що забезпечують масштабованість, координацію та спільне використання знань між функціональними акторами.: це – Агент Клонування (**СА**), Агент Координації (**СОА**) з його Простором Спільних Знань (SDS) і Агент Онтології (**ОА**).

Роль **СА** в даній задачі - клонувати Агентів Викладачів (**ТА**)кожний раз, коли агент-посередник "створює" нове завдання - опрацювати запит нового пошукача.

Сценарій прийому аспіранта: Сценарій прийому пошукача в аспірантуру було трохи адаптовано, щоб підкреслити переваги, які можна отримати, використовуючи підхід, поданий в секції 2. Були зроблені такі припущення: усі учасники процесу прийому - пошукачі, професори, та ін. находяться на онлайновому зв'язку під час виконання

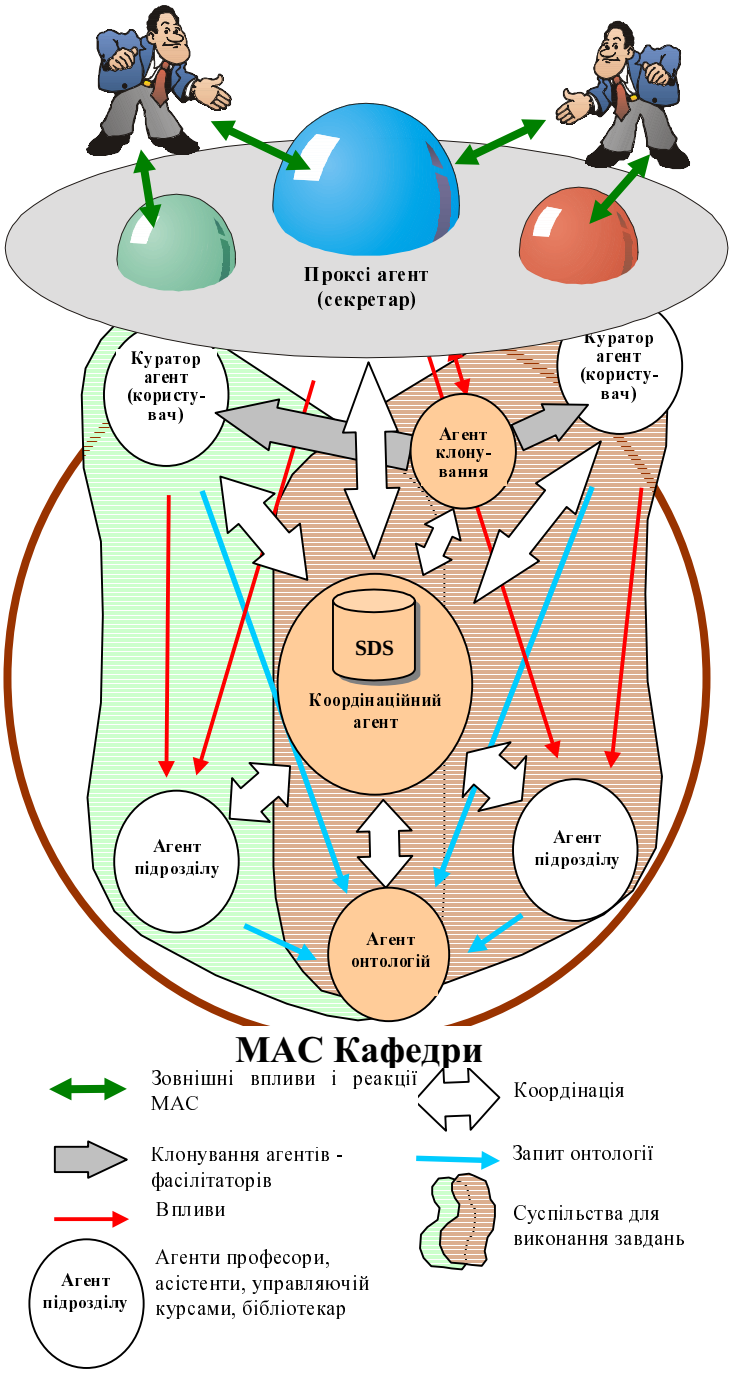

Рисунок 3.12 Структура МАС віртуальної кафедри

всього сценарію, а також всі потрібні роботи виконуються за короткий час.

Ми вважаємо, що процедура прийому в аспірантуру складається із таких етапів:

- Пошукач надсилає своє резюме і підтверджує своє бажання навчатися у аспірантурі
- Резюме аналізується і знаходиться Професор, який більш за інших підходить цьому пошукачеві
- Пошукач складає тести, підготовлені обраним Професором
- Цей пошукач, що успішно склав тести, проходить інтерв'ю і приймається до роботи над дослідницьким проектом
- Професор і його асистент готують індивідуальний графік роботи аспіранта, а також, перелік літератури, що йому необхідно прочитати.

Діяльності агентів в цьому процесі можуть бути такими:

Етап 1. В*становити зв'язок і прийняти резюме.*:

PA приймає зовнішній вплив s генерує нове завдання. Перші атомарні роботи в рамках цього завдання такі: **CA** – клонувати Агента Викладача; **PA** – перенаправити зв'язок пошукача із факультетом на зв'язок із конкретним **TA**, **TA** – запросити резюме і виділити дані про кваліфікацію пошукача

Етап 2. *Аналіз резюме і пошук найкращого Професора - керівника*:

TA віддає дані про кваліфікацію пошукача **PRA факультету**. **PRA** відповідають, формуючи параметричний відгук, згідно з параметрами кваліфікації. **TA** знаходить найкращій результат, у тому випадку, якщо параметри відповідей професорів знаходяться в межах необхідних. В тому випадку, якщо рівень кандидата не підходить, **TA** генерує завдання для Агента-Посередника відповісти пошукачеві і рекомендувати йому провести таку саму роботу на іншому факультеті. В тому випадку, якщо рівень кандидата є достатньо високим, кандидата кваліфікують і **TA** викликає новий етап – етап тестування.

Етап 3. *Тестування:*

TA запитує питання для тестів у **PRA**. **PRA** дає такі питання. **TA** пропонує пошукачеві відповісти на питання тесту і передає результати тестів **PRA**. **PRA** оцінює завдання і відповідає, ставлячи відмітки. **TA** проводить аналіз оцінок (подібно до етапу 2) і або кваліфікує пошукача і ініціює етап інтерв'ю, або просить **PA** повідомити пошукача про невдачу.

Етап 4. *Інтерв'ю:*

TA генерує завдання для **PRA** провести інтерв'ю з пошукачем. **TA** перенаправляє пошукача до **PRA**. **PRA** встановлює онлайновий зв'язок між своїм господарем (професором – людиною) і пошукачем. **PRA** вимагає від свого господаря – професора заповнити форму прийому в аспірантуру. **PRA** просить **TA** обробити цю форму із заповненими даними. **TA** аналізує форму і або передає її **PA** Відділу Кадрів для прийняття пошукача в аспірантуру, або просить **PA** повідомити пошукачу про невдалі результати інтерв'ю. Якщо пошукач успішно пройшов етап інтерв'ю, і тому став аспірантом, **TA** запускає етап створення індивідуальної програми.

Етап 5. *Розробка індивідуальної програми:*

TA генерує завдання для **PRA** приготувати індивідуальну програми роботи аспіранта на 1-ій семестр. **PRA** перенаправляє цей завдання своєму помічнику **AA**, щоб той добавив рекомендації щодо курсів у список параметрів. **AA** готує план роботи і запитує необхідні електронні курси у **CMA**. **CMA** аналізує запит, і, якщо треба, робить запит про додаткові, але ще недоступні, курси – приклад див. у 25.

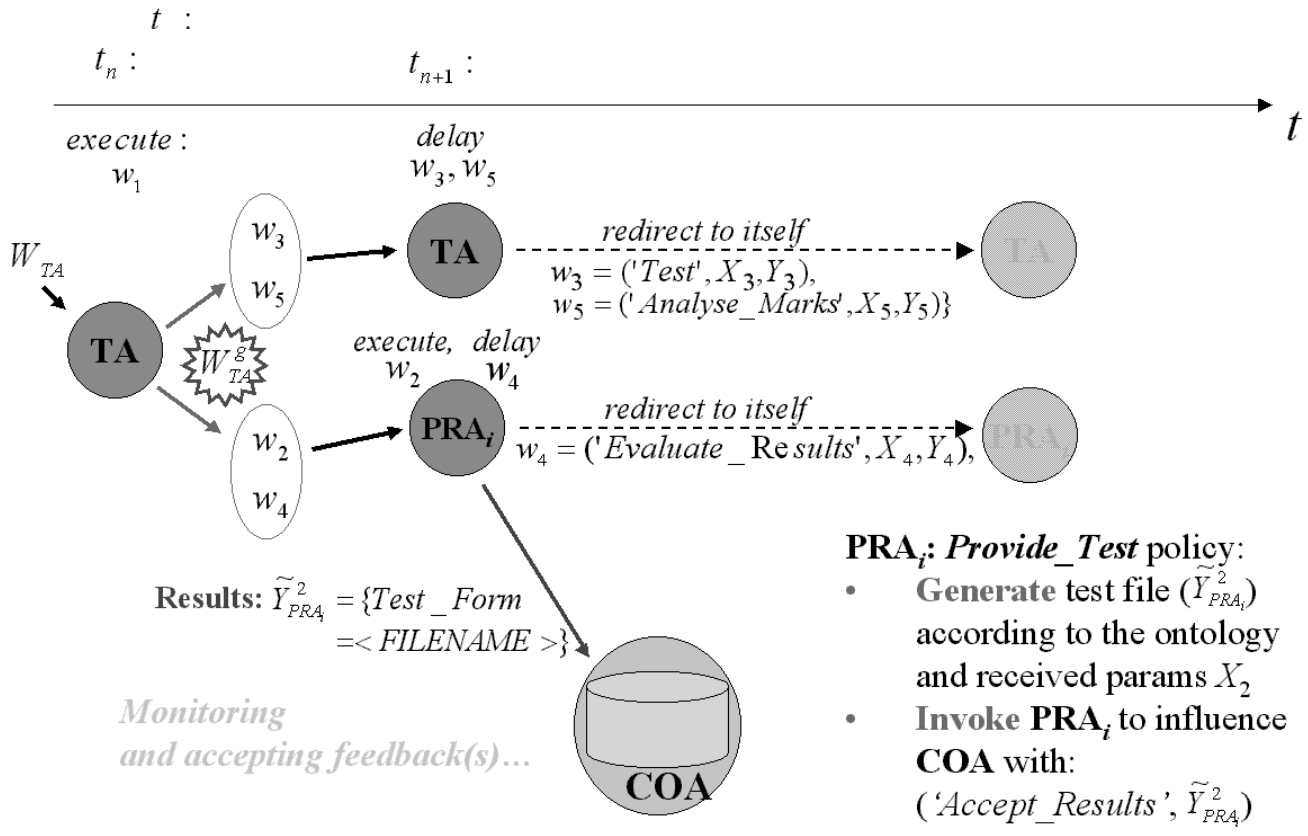

Рисунок 3.14. Тестування, фаза 2, представлення тесту (приведено по [54]).

Моделювання етапу тестування: Нехай в момент $t = t_n$ **TA** ініціалізує етап тестування $t = t_n$. Розглянемо діяльності агентів **TA**, **PRA** і **PA** на цьому етапі.

В момент $t = t_n$ (див. Рис. 3.13) **TA** приймає таку множину робіт

$W_{TA} = \{ w_1 = ('Require\ the\ test', X_1, Y_1)\}$

з такими параметрами і результатами роботи w_1:

$X_1 = \{$***Edu_Rating*** $=<$ *structure*, **ontology** $=$ *Edu_Rating* $>$,

Q_E_Rating $=<$ *structure*, **ontology** $=$ *Qualif_Exp_Rating* $>$,

Pub_Rating $=<$ *structure*, **ontology** $=$ *Publication_Rating* $>$,

Professor $=<$ *Id*, **ontology** $=$ *Agent_Name* $>\}$,

$Y_1 = \varnothing$.

Атомарна робота w_1 приймається і виконується, оскільки всі параметри X_1 (отримані як результати виконання робіт на попередніх етапах) є доступними через SDS **агента COA**. В процесі виконання роботи w_1 агент **TA** генерує завдання $\tilde{W}_{TA} = \{W^d_{TA}, W^g_{TA}\}$, де:

$W^d_{TA} = \varnothing$ тому що робота w_1 вже виконається і ніяких інших робіт не залишилося для передачі їх іншим агентам;

$W^g_{TA} = \{ w_2 = ('Provide\_Test', X_2, Y_2),$

$w_3 = ('Test', X_3, Y_3),$

$w_4 = ('Evaluate\_Results', X_4, Y_4),$

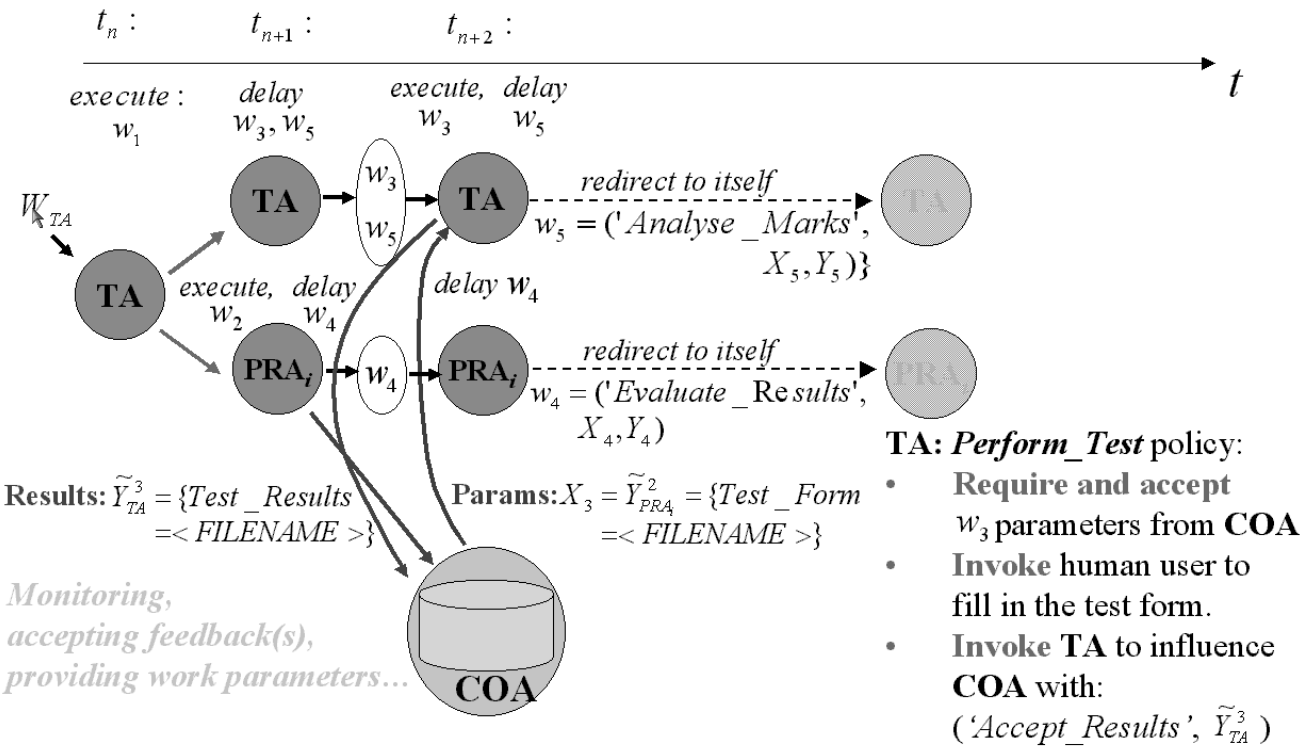

Рисунок 3.15. Тестування, фаза 3, виконання тесту (приведено по [54]).

$w_5 = ('Analyse\_Marks', X_5, Y_5)\}$.

Роботи w_2, w_4 сформують завдання W_{PRA} , а роботи w_3, w_5 сформують завдання W_{TA} в наступний момент часу $t = t_{n+1}$.

В момент $t = t_{n+1}$ (див. Рис. 3.14) **PRA** приймає

$W_{PRA} = \{ w_2 = ('Provide\_test', X_2, Y_2),$

$w_4 = ('Evaluate\_Results', X_4, Y_4)\}$.

Робота w_2 виконується і результати

$\tilde{Y}_2 = \{Test\_Form =< FILENAME >\}$

передаються **COA** для подальшого використання. Тим часом, робота w_4 відправляється **PRA** на наступний момент часу - результати роботи w_3, які формують параметри роботи w_4, ще не повернулися від **COA**. Тим часом, **TA** приймає

$W_{TA} = \{ w_3 = ('Test', X_3, Y_3),$

$w_5 = ('Analyse\_Marks', X_5, Y_5)\}$

і перенаправляє обидві роботи собі на наступні проміжки часу, чекаючи на результати робіт w_2, w_4. В момент $t = t_{n+2}$ (див. Рис. 3.15) **TA** виконує w_3. В момент $t = t_{n+3}$ **PRA** виконує w_4 і передає вектор результату $\tilde{Y}_{PRA} = \{\tilde{y}_4^1 = (m_1, m_2, ..., m_k), \tilde{y}_4^2 = (s_1, s_2, ..., s_k)\}$ агенту **COA**. k - це кількість проектів, якими керує господар агента **PRA**, де потрібні аспіранти, m_i - це оцінка

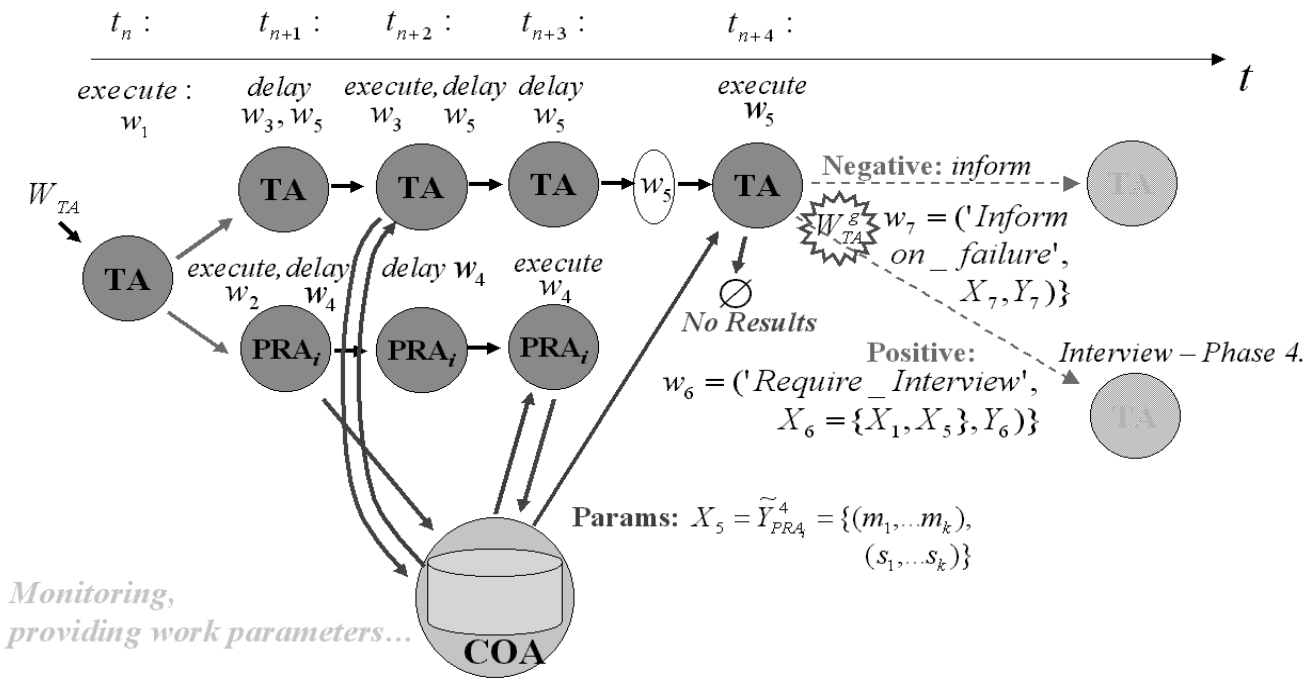

Рисунок 3.16. Тестування, фаза 5 - остання, аналіз параметрізованих оцінок (приведено по [54]).

пошукача, якщо він претендує на роботу над проектом *i*, та s_i відображає найнижчу (за думкою професора) оцінку необхідного рівня пошукача.

В момент $t = t_{n+4}$ (див. Рис. 3.16) **TA** приймає

$W_{TA} = \{w_5 = ('Analyse\_Marks', X_5, Y_5)\}$,

де

$X_5 = \{$***Marks*** $=< \tilde{y}^1_4,$ **ontology** $=$ *Mark_per_Project* $>$,

Scale $=< \tilde{y}^2_4,$ **ontology** $=$ *Positive_Mark_per_Project* $>\}$

і вирішує, чи може даний пошукач отримати місце в тому чи іншому проекті. Якщо це можливо, W^g_{TA} буде складатися із роботи $w_6 = ('Require\_the\_Interview', X_6 = \{X_1, X_5\}, Y_6)$,

в інших випадках **TA** генерує роботу

3.3 **Висновки**

Розглянуті вище приклади моделювання реальних функціональних систем показують доцільність пропонованого підходу до моделювання складних функціональних систем.

Висновки

Проблема моделювання складних функціональних систем досить актуальна. Основною особливістю моделювання подібних функціональних систем є необхідність розробки і побудови моделі системи в термінах інформаційної взаємодії компонент. При цьому проблема може бути декомпозирована на задачу моделювання конкретного компонента системи і задачу побудови загальної моделі системи в цілому. Така декомпозиція дозволяє зосередити сили розроблювачів математичних моделей на рішення більш простих задач. Істотною особливістю математичного моделювання складних функціональних систем так само є досить специфічна вимога до вибору режимних параметрів окремих компонет системи. У загальному випадку, кожний компонент системи досить просто описується вербально, але подібне описане не можливо використовувати для формування загальної математичної моделі системи в цілому.

Для рішення відзначених проблем, що виникають у силу особливостей моделювання складних інформаційних систем, доцільне застосування диакоптического підходу і його розвитку, відомого як принципи макромоделювання.

Основним результатом роботи є демонстрація можливості застосування принципу макромоделювання для організації математичної моделі складної функціональної системи в термінах інформаційної взаємодії компонент і функціонування системи в цілому.

У результаті проведеного дослідження удалося раціонально декомпозировать досить складну задачу, виділити доцільні для інформаційної моделі режимні параметри, розробити формальні методи організації математичної моделі системи в цілому і формальні принципи організації макромоделі – об'єкта системи.

Результати роботи в 2000 р. апробовані на:

- міжнародній Науково - практичній конференцфї із програмування УкрПРОГ'2000, Кіев;
- міжнародній Науковій конференції "Інтелектуальна обробка інформацфї" - ІОІ'2000, Алушта;
- міжнародном конгресі "Інформацфйне суспільство в Україні" - Конгрес'2000, Кіев;
- міжнародній Науковій конференції "Information Society of the XXI century: Emerging technologies and New Challenges" - IS'2000, Аізу-Вакамацу, Япония.

Перелік посилань

1. Ермолаев В.А., Плецкий С.Ю., Толок В.А. Архитектура унифицированного информационного пространства виртуального университета // Вісник Запорізького державного університету. - 1998, Т. 1, № 2, с. 44-53, ISBN 966-599-007-1.
2. Anghern, A Designing Mature Internet Business Strategies: The ICDT Model.// European Management Journal, Vol. 15, No 4, 1997, pp. 361-369.
3. I3Net European Research Initiative http://www.i3net.org
4. Papazoglou, M. P., Van der Heuvel, W.-J. From Business Processes to Cooperative Information Systems: An Information Agents perspective.//In: M. Klusch (Ed.) Intelligent Information Agents: Agent Based Information Discovery and Management on the Internet. Springer-Verlag, 1999, pp. 10-36.
5. Davulcu, H., Kifer, M., Pokorny, L. R., Ramakrishnan, C. R., Ramakrishnan I. V. Modeling and Analysis of Interactions in Virtual Enterprises. //In: Proc. of the 9-th International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises (RIDE-VE'99), March 23-24, 1999, Sidney, Australia.
6. OMG Unified Modeling Language. Specification. Version 1.3. June 1999. http://www.rational.com/uml/resources/documentation/
7. Lupu, E., Milosevic, Z., Sloman, M. Use of Roles and Policies for Specifying, Building and Managing Virtual Enterprise. //In: Proc. of the 9-th International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises (RIDE-VE'99) , March 23-24, 1999, Sidney, Australia.
8. Jennings, N. R., Faratin, P., Johnson, M. J., Norman, T. J., O'Brien, P., Wiegand, M. E. Agent-based business process management.// Int. Journal of Cooperative Information Systems, 5(2, 3), pp. 105-130.
9. Sycara, K., Decker, K., Pannu, A. Williamson, M. and Zeng, D. Distributed Intelligent Agents. // IEEE Expert, Dec. 1996, pp. 36-45.
10. V. A. Ermolayev, S. U. Borue, V. A. Tolok, N. G. Keberle, Use of Diakoptics and Finite Automata for Modelling Virtual Information Space Agent Societies // "Lecture Notes of Zaporozhye State University", ISBN 966-599-058-4, Vol. 3, No 1, 2000, pp. 34-44.
11. Борю С.Ю., Ермолаев В.А., Толок В.А. О диакоптическом подходе к моделированию процессов во многофункциональных информационных системах // (Спец. выпуск) Доклады международной конференции KDS'99, Кацивели, 13-18.09.1999 / Искусственный интеллект №2, 1999, с. 211-219, ISSN 1561-5359.
12. V. A. Ermolayev, V. A. Tolok, Interfaces and Human - Agent Interaction in Virtual Information Spaces, Panel talk at ESPIRIT AgentLink SIG on Intelligent Information Agents meeting, London, GB, Apr. 21-23, 1999, 10 p.
13. Бутырин, П. А., Борю, С. Ю. Комплекс программ макромоделирования систем // Сб. Моделирование силовых вентильных преобразователей., -К.: ИЭД, -1989, с. 12-20.
14. Kron, G., Diacoptics. Macdonald, London, 1963
15. Nwana, H. S. Software Agents: an Overview. // Knowledge Engineering Review, Vol. 11, No 3, pp. 205-244, Oct./Nov. 1996.
16. Baral, C., Lobo, J., Formalizing Workflows as Cooperative Agents In Proc. of DYNAMICS 97 (a workshop in ILPS 97).
17. Adam, N., Atluri, V. and Huang, W. Modeling and analysis of workflows using petri nets. // Journal of Intelligent Information Systems, 10(2), pp. 131-158, March 1998.
18. Sycara, K. In-Context Information Management through Adaptive Collaboration of Intelligent Agents. .//In: M. Klusch (Ed.) Intelligent Information Agents: Agent Based Information Discovery and Management on the Internet. Springer-Verlag, 1999, pp. 78-99.
19. Foundation for Intelligent Physical Agents (FIPA) Spec: DRAFT, Version 0.2, Agent Communication Language, 1999, http://www.fipa/org

20. Rao, A. S. and Georgeff, M. P. Modeling rational agents within a BDI-architecture. In Fikes, R. and Sandewall, E., editors, *Proceedings of Knowledge Representation and Reasoning (KR&R-91*), 1991, pages 473–484. Morgan Kaufmann Publishers: San Mateo, CA.
21. Yannis Labrou, Tim Finin, and Yun Peng: "Agent Communication Languages: The Current Landscape"
22. Object Managament Group, “The Common Object Request Broker: Architecture and Specifications”, OMG Document Number 91.12.1, December 1991.
23. Finin, T. and Fritszon, R. KQML - A language for protocol and information exchange. // Proc 13th DAI workshop, pp. 127-136, Seattle, WA, USA.
24. Hyacinth Nwana, Lyndon Lee, and Nick Jennings: "Coordination in Software Agent Systems", BT Technology Journal, 14(4), 1996, pp 79-88.
25. S. U. Borue, V. A. Ermolayev, V. A. Tolok: Application of Diakoptical MAS Framework to Planning Process Modelling // in: "Problems of Programming" Scientific Journal №1-2, 2000, ISBN 966-02-1244-5, Special Issue: Proc. of the 2-nd Intl. Scientific - Practical Conference on Programming (UkrPROG'2000), Kiev, 23-26 May 2000, pp. 488-500.
26. Genesereth M.R., Fikes R.E., et.al. Knowledge Interchange Format Version 3.0 Reference Manual. Logic-92-1, Stanford University Logic Group, 1992.
27. Vadim Ermolayev: Dynamic Agent Communities Facilitating to Distant Learning in a Virtual University Information Space. // Proc. of Intl. Conf. Information Society in the 21st Century (IS2000), Spec Session on Virtual Universities and Distant Educartion (VUDE), Aizy-Wakamatsu, Japan, Nov. 5-8, 2000, pp. 488-495.
28. Adali S., Subrahmanian V.S. Amalgamating knowledge bases, II. Distributed mediators. International Journal of Intelligent and Cooperative Information Systems 3(4): 349-383, 1994.
29. Adiba M.et al. POLYPHEME:An Experience in Distributed Data Base System Design and Implementation.-In: Proc. of the International Symposium on Distributed Data Bases. Paris. Amsterdam: North-Holland, 1980.
30. Arens Y., Knoblock C.A., Shen W. Query Reformulation for Dynamic Information Integration. Journal of Intelligent Information Systems. 1996.
31. Bayardo et al. InfoSleuth: Semantic Integration of Information in Open and Dynamic Environment. In Proceedings of the 1997 ACM International Conference on the Management of Data (SIGMOD), Tucson, Arisona, May 1997.
32. Benjamins V.R. et al. IBROW3: An Intelligent Brokering Service for Knowledge-Component Reuse on the World – Wide Web. Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling and Management, KAW’98.
33. Cardenas A.F, Pirahesh M.H. Data Base Communication in a heterogeneous data base management system network. -Information Systems, 1980, 5, p.55-79.
34. Garcia-Molino H. et. Al. The TSIMMIS Approach to Mediation: Data Models and Languages. In Proceedings of the NGITS (Next Generation Information Technologies and Systems), June 1995.
35. Gruber T. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5:199-220, 1993.
36. Guarino N., Masolo C., Vetere G. Content-Based Access to the Web. IEEE Intelligent Systems, May/June 1999, p.70-80.
37. Guarino N. The Role of Ontologies in Information Systems Design. Proceedings of the First International Conference on Formal Ontologies, FOIS’98.
38. Guarino N., The Ontological Level. In: Casati R., Smith N. and White G.(eds.), Philosophy and the Cognitive Sciences, Vienna: Ho:lder-Pichler-Tempsky, 1994.
39. Lenat D. et al. CYC: Toward programs with Common Sense, Communications of the ACM, Vol.33, No.8, august 1990, p. 30-49.
40. Levy A., Srivastara D., Kirk T. Data Model and Query Evaluation in Global Information Systems, Journal of Intelligent Information Systems, 5(2), September 1995.

41. Lu J., Nerode A., Subrahmanian V.S. Hybrid Knowledge Bases, IEEE Transactions on Knowledge and Data Engineering, 1994.
42. Mena E., Kashyap V., Sheth A., Illaramendi A. OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-Existing Ontologies. In Proceedings of the First IFCIS International Conference on Cooperative Information Systems (CoopIS'96), Brussels (Belgium), June. IEEE Computer Society Press, 1996.
43. Object Managament Group, "Object Managament Architecture Guide", OMG Document Number 91.11.1, September 1, 1992.
44. Object Managament Group, "Object Services Architecture", Revision 8.0.
45. Sheth A.P. Changing Focus on Interoperability in Information Systems: from System, Syntax, Structure to Semantics. In: Interoperating Geographic Information Systems. Goodchild M.F., Egenhofer M.J., Fegeas R. and Kottman C.A. (eds.). Kluver. 1998.
46. Sowa J. Conseptual Structures: Information processing in Mind and Machine. Addison-Wesley, Reading, Mass.,1984.
47. Uschold M., Gruninger M. Ontologies: Principles, Methods and Applications. Knowledge Engineering Review, 11(2), 1996.
48. Брюхов Д.О., Задорожный В.И., Калиниченко Л.А., Курошев М.Ю., Шумилов С.С. Интероперабельные информационные системы: архитектуры и технологии. // СУБД. Москва, 1995, №4. – С.86-113.
49. Калиниченко Л.А., Рывкин В.М., Чабан И.А. Принципы организации и архитектура СИЗИФ - системы организации интегрированных баз данных // Программирование. – Москва, 1975, № 4.
50. Калиниченко Л.А., Рывкин В.М., Чабан И.А. Основные особенности языка манипулирования данными в системе интегрированного запоминания информации СИЗИФ // Программирование. – Москва, 1975, № 6.
51. Калиниченко Л.А. 1983. Методы и средства интеграции неоднородных баз данных. – Москва: Финансы и Статистика, 1983. – 300 с.
52. Программа исследований в области баз данных на следующее десятилетие (Асиломарский отчет о направлениях исследований в области баз данных) // Открытые системы. – Москва, 1999, №1.
53. N. R. Jennings, P. Faratin, M. J. Johnson, P. O'Brien, M. E. Wiegand: "Using Intelligent Agents to Manage Business Processes", Proc. First Int. Conf. on The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM96),pp. 345-360. London, UK.
54. Vadim Ermolayev: Dynamic Agent Communities Facilitating to Distant Learning in a Virtual University Information Space (the transparences) http://eva.zsu.zaporizhzhe.ua:80/eva_personal/PS/vude-slides.pdf