Київський національний торговельно-економічний університет

В.Л. Плескач,
Ю.В. Рогушина

Агентні технології

Монографія

Київ 2005

ББК З.973
УДК 004.738.5

Плескач В.Л. , Рогушина Ю.В.

Агентні технології: Монографія. – К.: Київ. нац. торг.–екон. ун–т, 2005. – 344 с. (Табл.10. Рис.58. Бібліограф.361)

У монографії розглянуто теоретичні основи та застосування агентних технологій. Висвітлено сучасні підходи до подання й обробки знань, на яких базуються інтелектуальні програмні агенти. Наведено моделі та технології створення програмних агентів і мультиагентних систем, їх застосування для пошуку інформації та підтримки електронного бізнесу.

Для науковців і спеціалістів, які займаються дослідженнями та розробками в галузі інтелектуальних інформаційних систем, бізнес-застосунків інформаційної економіки.

ББК З.973
УДК 004.738.5

Рецензенти: О.Л. Перевозчикова, чл.-кор. НАНУ, д-р фіз.-мат. наук, проф.,
А.І. Нікітін, д-р техн. наук, проф.,
О.А. Корольов, д-р екон. наук, проф.

Рекомендовано вченою радою Київського національного торговельно-економічного університету
(протокол №3 від 20 січня 2005 р.)

ISBN 966–629–163–0

Зміст

Вступ

Тим, хто нас народив, присвячується

Сучасний етап розвитку інформаційних технологій характеризується впровадженням систем, що базуються на знаннях. Існує багато підходів до аналізу інтелектуальних агентів та мультиагентних систем. Ця проблематика має багаторічну історію. Актуальність теми визначається широким спектром застосувань агентних технологій. Значна кількість сучасних програмних розробок у сфері розподіленого штучного інтелекту пов'язана саме із застосуванням агентів.

Агентно-орієнтоване програмування набуло останнім часом великої популярності. Інтелектуальні програмні агенти – це новий клас програмного забезпечення. Вони є новим рівнем абстракції, що дозволяє створювати розширювані, масштабовані, інтероперабельні системи. Програмному агенту делегуються певні функції людини, часом досить складні, для надання користувачеві потрібних йому послуг. Моделювання поведінки агентів і мультиагентних систем потребує глибокого теоретичного обґрунтування. На основі дослідження моделей та архітектур інтелектуальних агентів запропоновано формальну теоретичну модель, яка для опису поведінки агента використовує інтенсіональні оператори.

Якщо користувач має модель переконань, знань, намірів і цілей системи, то він розуміє, які дії система вибирає в різних ситуаціях. Це розв’язує соціальну проблему застосування агентів: люди хочуть використовувати програмне забезпечення з прогнозованою поведінкою і керованими діями.

Значна увага у роботі приділяється моделям та мовам обміну знаннями між агентами. Приділяється увага застосуванню агентних технологій в інформаційній економіці. Глави 1, 2 та 4 розроблені авторами спільно, глава 3 – Рогушиной Ю.В.

У главі 1 розглянуто інтелектуальні автоматизовані системи і їх властивості, моделі подання знань в цих системах, засоби подання та обробки онтологій, логічні методи здобуття знань та використання методів штучного інтелекту в експертних системах. Онтології сьогодні є базисом для інтероперабельного подання знань, якими користуються інтелектуальні програмні агенти. Наведені у цій главі оригінальні методи індуктивного узагальнення дозволяють агентам навчатися за власним досвідом.

У главі 2 розглянуті теоретичні основи інтелектуальних програмних агентів, їх основні властивості та таксономії, мультиагентні системи, агентно-орієнтоване програмування, засоби взаємодії програмних агентів, методології їх створення та підходи до підвищення їх інтелектуальності. Докладно проаналізовано використання інтенсіональних відношень для моделювання поведінки інтелектуальних агентів та мультиагентних систем. Запропоновано формальний апарат для опису поведінки агентів, пов'язаний з модальними логіками та семантикою можливих світів.

Глава 3 присвячена застосуванню агентних технологій для пошуку інформації в Інтернеті. У цій главі проаналізовано типи інформаційних ресурсів Інтернету, засоби формального подання метазнань про них та методи їх пошуку. Розглянуті інформаційно-пошукові системи: машини пошуку та каталоги, метапошукові системи. Досліджуються інформаційно-пошукові агенти та мультиагентні інформаційно-пошукові системи. Запропонована архітектура мультиагентної інформаційно-пошукової системи, яка використовує онтологічне подання знань користувача про область його інтересів для підвищення релевантності пошуку. Агенти, що входять до складу цієї системи, описані через інтенсіональні відношення. Значна увага приділяється інформаційно-пошуковим агентам через те, що вони є компонентами багатьох спеціалізованих мультиагентних систем, в яких пошук інформації в розподіленому середовищі є однією з складових.

У главі 4 розглянуто використання агентних технологій в сфері електронного бізнесу: електронній комерції, логістиці, електронному урядуванні, дистанційній освіті, електронній медицині тощо. Проаналізована інформаційна економіка як основна сфера застосування агентних технологій.

Основна ціль монографії – це створення теоретичної основи для розробки та використання інтелектуальних агентно-орієнтованих систем. Наведені у роботі приклади застосування програмних агентів та мультиагентних систем мають показати ефективність агентного підходу та підкреслити сферу його застосуванню.

Розглянуті архітектурні рішення та методики розробки агентів мають допомогти розробникам у створенні бізнес-застосунків, призначених для рішення конкретних задач.

Глава 1. Системи, засновані на знаннях

Знання – це сила
Ф. Бекон

Сучасний етап розвитку інформаційних технологій (ІТ) характеризується впровадженням систем, що базуються на знаннях (СБЗ). Це новий якісний етап в еволюції систем обробки інформації. СБЗ обробляють дані, що мають складну структуру, використовуючи потенціал практичного досвіду людей.

СБЗ відрізняються тим, що вони орієнтовані на рішення задач, які важко формалізуються. Розробникам СБЗ не відоме алгоритмічне рішення задачі (хоча, можливо, воно й існує), а мета роботи СБЗ не може бути подана в термінах точно визначеної цільової функції. Такі системи корисні там, де наука не може створити конструктивних рішень, область визначень міняється, ситуація залежить від контекстів і мовна (описова) модель домінує над алгоритмічною, плани рішення задач невідомі.

1.1. Інтелектуальні системи: основні визначення та проблеми

Підвищення ефективності впровадження інформаційних систем (ІС) у різні галузі людської діяльності безпосередньо пов'язано з рівнем інтелектуалізації цих ІС. ІС – сукупність апаратно-програмних засобів обробки інформації в певній предметній області (ПрО) для досягнення поставленої користувачем мети. На сучасному етапі розвитку ІТ більшість рутинних операцій з перетворення інформації вже автоматизовано і подальше збільшення ефективності роботи потребує автоматизації інтелектуальної, творчої діяльності людини. Інтелектуальні автоматизовані системи (ІАС) складається з таких компонентів:

- вирішувача;
- інформаційного середовища;
- інтелектуального інтерфейсу.

Інформаційне середовище складається з бази фактів і бази знань. Інтелектуальний інтерфейс забезпечує діалог з користувачем, демонстрація результатів роботи ІАС та навчання роботі з системою.

Для визначення поняття "*інтелектуальна автоматизована система*" (ІАС) необхідно викласти тлумачення терміну “інтелект”.

Інтелектом (від латин. “*intellectus”* – "пізнання") називають здатність міркувати, діяти цілеспрямовано, правильно реагувати на ситуацію. Відповідно, *інтелектуальними* є задачі, для рішення яких

немає чітко заданого алгоритму, що завжди приводить до потрібного результату, а *інтелектуальною діяльністю* – процес рішення інтелектуальних задач.

Інтелектуальним задачам властиві неповнота, неточність і суперечливість знань ПрО, а також велика розмірність простору рішень, що не дозволяє розв'язувати їх простим перебором. У таких задачах часто немає чітких критеріїв для вибору оптимального рішення, а сама задача не завжди цілком формалізується. Прикладом інтелектуальної задачі є розпізнавання образів, тобто визначення належності об'єкта, що спостерігається, до однієї із заздалегідь визначених категорій. Основні властивості інтелектуальних задач:

- символьне подання даних;
- відсутність строгої постановки задачі;
- відсутність прийнятного для практичного використання алгоритму рішення;
- неповнота, неточність і суперечливість знань ПрО;
- відсутність чітких критеріїв вибору оптимального рішення;
- велика розмірність простору рішень.

Інтелектуальна діяльність – це дії людей, що приводять до отримання бажаного результату в ситуаціях, коли алгоритм вирішення проблеми відсутній. Іншими словами, це процес одержання бажаного результату в інтелектуальних задачах. Людина володіє певним набором знань про навколишній світ, які дозволяють їй орієнтуватися в різних ситуаціях і приймати правильні рішення. Крім того, людина вміє використовувати ці знання.

У зв'язку з цим потрібно розглянути термін "алгоритм". Поняття *алгоритму* є базовим для всіх галузей комп'ютерного програмування. Термін "алгоритм" ("algorithm") подається у виданні словника Webster's New World Dictionary, що вийшов у 1957 р., правда, дещо в іншому звучанні "algorism" – стародавнє слово, що позначає "виконання арифметичних операцій за допомогою арабських цифр" і походить від імені автора знаменитого перського підручника з математики IX сторіччя Мухаммеда аль-Хорезмі.

Алгоритм – це метод, якому властиві такі ознаки

- *скінченність* – закінчення роботи за скінченну кількість кроків;
- *визначеність* – дії, що потрібно виконати, строго й однозначно визначені для всіх можливих ситуацій;
- *наявність вхідних даних* – дані, з яких починається робота алгоритму;

- *наявність результуючих даних* – дані, що формуються внаслідок виконання алгоритму;
- *ефективність* – здатність алгоритму перетворювати вхідні дані в результат.

Можна визначити такі основні параметри алгоритму: :

- сукупність можливих вхідних даних;
- сукупність можливих результуючих даних;
- сукупність можливих проміжних даних;
- правило початку;
- правило безпосереднього перетворення;
- правило закінчення;
- правило отримання результату.

Знаходження алгоритмів є основною метою людини при розв'язку різноманітних класів задач. Відшукання алгоритму для задач певного типу пов'язано зі складними міркуваннями, що вимагають участі інтелекту людини. Доказом еквівалентності різних класів алгоритмів займалися такі вчені, як Пост, Тьюринг, Марков, Колмогоров. Процес вирішення задач, для яких знайдені відповідні алгоритми, не потребує інтелектуальних зусиль і тому його може здійснювати об'єкт (людина або комп'ютер), здатний виконувати елементарні операції, з яких складається алгоритм.

*Квазіалгоритми***.** Автоматизована ІС – програмна реалізація конкретного алгоритму. Інтелектуальні ІС – реалізація алгоритму, який не існує або нам не відомий. На перший погляд, маємо протиріччя. Проте це не так. Можна запрограмувати не безпосередньо сам алгоритм, а засоби, за допомогою яких ІАС автономно за прикладами навчиться цьому алгоритму (зокрема, цей прийом часто застосовують при розробці програмних агентів та нейромереж). Якщо ж алгоритм рішення задачі є надто складним, то можна реалізувати його спрощений варіант, який дозволяє отримати рішення з точністю, задовільною для практичного застосування.

Узагальненням поняття алгоритму є *квазіалгоритм*. На відміну від алгоритму інструкції квазіалгоритму можуть бути не зовсім чіткими, тому результат квазіалгоритму не може бути гарантованим.

Можлива також ситуація, коли алгоритм рішення задачі взагалі відсутній. У цьому випадку його можуть замінити запропоновані користувачем метазнання у вигляді набору евристик, які дозволяють знайти прийнятне (хоча, можливо, не оптимальне) рішення для певної підмножини вхідних даних. Цей шлях найближчий до способу прийняття рішень, який звичайно застосовує людина за відсутності

чітких знань і досвіду рішення схожих задач для виконання будь-яких дій, що призводять до зміни ситуації.

Аналіз підходів до оцінки рівня інтелектуальності ІС

Інтелектуалізація ІС – це процес підвищення рівня інтелекту ІС. Пропонуємо наступне визначення.

Нехай є дві ІС – А та В, здатні знаходити рішення $R_A(d)$ і $R_B(d)$ відповідно в ситуаціях d з визначеної ПрО D. Система В отримана внаслідок інтелектуалізації А, $B = Int(A)$, якщо:

- система В завжди одержує рішення в ситуаціях, у яких його знаходить і система А, $\exists R_A(d) \Rightarrow \exists R_B(d)$;
- існують ситуації, у яких А не знаходить рішення, а В – знаходить.

Від рівня інтелектуальності ІС залежить, на скільки кроків система здатна прогнозувати свою поведінку і реакцію середовища на свої дії (приміром, гросмейстер може оцінити наслідки своїх рішень на багато ходів, а шахіст-початківець – у кращому випадку на 1–2 ходи). На рівень інтелектуальності системи впливають як здатність навчатися і самонавчатися, тобто використовувати наявні знання в нових заздалегідь невідомих ситуаціях, так і широта її ПрО [258]. Поняття інтелектуалізації ІС містить у собі такі аспекти, як підвищення "грамотності" ІС, наявність метазнань ПрО і розширення способів одержання нових знань [247]. Чим вище інтелектуальність ІС, тим ефективніше вона може досягати поставлену ціль [275]. Нові знання формуються шляхом застосування наявних знань – інформації про процеси рішення, логічне виведення, закономірності ПрО тощо – до даних, з яких породжується нова інформація [272]. Однак такі властивості, як «широта ПрО» і «нові знання», складно оцінити кількісно.

Висування гіпотез і узагальнення емпіричних фактів як вид інтелектуальної діяльності людини відіграє найважливішу роль у процесі пізнання і вважається однією з характерних властивостей людського розуму. Тому наявність в ІС механізмів моделювання недедуктивних міркувань, зокрема індуктивного узагальнення, і ступінь "досконалості" цих механізмів природно розглядати як один з показників інтелектуальності інформаційних систем. З формальної точки зору додання різних видів недедуктивного виведення в БЗ сприяє розширенню логічної формальної системи подання знань.

У [265] рівень інтелектуальності ІС $I(A) = S(s_1,...,s_9), i = \overline{1,9}$ визначається наявністю в неї таких властивостей:

- s_1 – автономність роботи;

- s_2 – взаємодія з іншими об'єктами;
- s_3 – сприймання інформації про навколишній світ;
- s_4 – застосування абстракції;
- s_5 – використання знання ПрО;
- s_6 – адаптивність поведінки;
- s_7 – здатність до навчання;
- s_8 – толерантність до помилок;
- s_9 – здатність до спілкування природною мовою.

Цей підхід дозволяє в деяких випадках порівнювати рівень інтелектуальності ІС, призначених для рішення задач у різних областях (якщо в системі А є всі властивості системи В, але в системі В є властивості, яких немає в системі А, тоді I(B) – рівень інтелектуальності системи В – вище, ніж у системи А):

$$\exists S(A) \subseteq S, S(B) \subseteq S, S(A) \subseteq S(B) \Rightarrow I(B) \geq I(A).$$

Проте значна частина ІС при такому підході непорівнянні: $\exists S(A) \subseteq S, S(B) \subseteq S, S(A) \not\subset S(B), S(B) \not\subset S(A) \Rightarrow I(B)$ і $I(A)$ непорівнянні. Наприклад, складно порівняти рівень інтелектуальності ІС, що здатна до спілкування природною мовою, але не здатна до навчання (різноманітні чат-імітатори), і систем індуктивного виведення нових знань, що не підтримують природномовний діалог. Крім того, усе наведене вище може бути властиве ІС у більшому або меншому ступені, а оцінити кількісно деякі неможливо.

У [302] пропонується кількісна оцінка рівня інтелектуальності ІС $I(A) = \sum_{i=1}^{6} s_i * w_i$, де w_i – вага і–ї властивості, а s_i – значення і–ї властивості:

- s_1 – використання моделі навколишнього світу для формування планів власних дій;
- s_2 – наявність альтернативних варіантів при плануванні дій;
- s_3 – здатність реконструювати план під час його виконання, якщо здійснювані дії призводять до небажаних наслідків;
- s_4 – використання власного досвіду для розширення і корекції моделі ПрО;
- s_5 – спілкування з користувачем природною мовою;
- s_6 – допустимість періоду виконання плану рішення задачі.

З цих показників користувач ІС може самостійно оцінити тільки два останніх, і йому треба цілком покладатися на твердження розробників системи, що не завжди є об'єктивними і коректними.

Пропонуємо критерій оцінки рівня інтелектуальності ІС $I(A) = f(T, X, L, C, P, K)$, завдяки якому завжди виконується умова (1): при інтелектуалізації якої-небудь ІС рівень її інтелектуальності має збільшитися (або хоча б не зменшитися).

Цей критерій використовує набір параметрів, значення яких користувач системи здатний сам оцінити кількісно (хоча б дуже приблизно) [308].

$$\exists f(T, X, L, C, P, K), \forall A, B, B = Int(A) \Rightarrow$$
$$\Rightarrow f(T(A), X(A), L(A), C(A), P(A), K(A)) \leq, \quad (1)$$
$$\leq f(T(B), X(B), L(B), C(B), P(B), K(B)),$$

де

- T – тип атомарних елементів;
- X – кількість атомарних елементів;
- L – кількість зв'язків між ними;
- C – кількість команд у нормалізованому алгоритмі;
- P – співвідношення кількості успішних експериментів з кількістю проведених;
- K – клас задач, які розв'язує ІС.

Складність структури даних, які здатна обробляти ІС, характеризують тип атомарних елементів T, їхня кількість X і кількість зв'язків між ними L. Складність методів перетворення даних характеризує кількість команд у нормалізованому алгоритмі C. Очевидно, що користувач не може точно оцінити ці параметри, але може дати їм приблизну якісну оцінку.

Імовірність того, що дії ІС приводять до мети, яку вона декларує, оцінюється як співвідношення кількості успішних експериментів з кількістю проведених P. Цей параметр користувач звичайно може оцінити досить точно.

$$I(A) = f(T, X, L, C, P, K) = g(T, X, L) * h(C) * P * \varphi(K). \quad (2)$$

Для простоти оцінки в якості g(T,X,L) ми використовуємо $g(T, X, L) = g_1(T) * g_2(X, L)$, де функції $g_1(T)$ і $g_2(X,L)$ задаються таблично. Таким чином,

$$I(A) = g_1(T) * g_2(X, L) * h(C) * P * \varphi(K). \quad (3)$$

Введемо шкалу інтервалів для значень параметрів: "*дуже мало*", "*мало*", "*середньо*", "*багато*", "*дуже багато*". Їх обробка базується на теорії нечітких множин, тобто мультиплікація двох нечітких значень визначається як добуток їхніх відносних ваг, а об'єднання – як їхня сума. $h(C)$ і $\varphi(K)$ також задаються таблично.

Ці оцінки задаються нами апріорі на підставі власного досвіду використання й аналізу ІТ у різних областях і вимагають подальшої деталізації, розширення й уточнення.

Якщо ІС обробляє дані різних типів, то

$$g_1(T_1,...,T_n)=\sum_{i=1}^{n} g_1(T_i). \qquad (4)$$

1.2. Символьні обчислення

АІС призначені для обробки інформації певної ПрО, тому закономірно, що вони задають її певну модель. У процесі розвитку ІТ прослідкуються два покоління, що відрізняються типами моделей ПрО та принципами їх автоматизованого оброблення.

До першого покоління належать системи, у яких моделлю ПрО є множина алгоритмів, на основі яких працює АІС. До недоліків цього підходу відносяться громіздкість таких моделей, неможливість їх стандартизації та повторного використання. Таку модель неможливо адаптувати до змін середовища та до нових задач без повної зміни ПЗ.

Для другого покоління АІС моделлю ПрО є знання системи про методи планування рішень. Основна частина ПЗ, що забезпечує планування рішень, обслуговування БД і БЗ, не орієнтована на конкретну ПрО. Незалежність ПЗ від розв'язуваних задач і специфіки конкретної ПрО дозволяє ввести єдину методику його розробки. Адаптація системи до змін середовищ і задач забезпечується змінами у БЗ та БД, а не в ПЗ, що їх обробляє.

При проектуванні систем другого покоління основна увага приділяється не розробці ПЗ, а проектуванню та заповненню БЗ. У зв'язку з цим особливого значення набувають засоби подання знань та методи їх автоматизованої обробки.

Формальні теорії

Формальні системи – аксіоматичні системи, що містять певну кількість заздалегідь обраних висловлень-аксіом.

Синтаксична структура понять ПрО звичайно описується *формальною теорією,* яка дозволяє за допомогою формальних методів отримувати нові твердження про властивості об'єктів або явищ ПрО, що є змістовною інтерпретацією виведених формул.

Побудова теорії певної галузі знань включає дослідження її структури та вибір позначень, які характеризують особливості цієї структури. Важливим інструментом є формальні теорії.

Формалізацією теорії називається її опис на базі строгої логіко-математичної мови.

Логіко-математична мова Ω задається набором з чотирьох множин Ω = <Srt, Cnst, Fn, Pr>, де

- Srt – непорожня множина, елементи якої називаються *сортами об'єктів*. Для кожного сорту $\pi \in$ Srt фіксується скінченний набір символів $u_1^{\pi}, u_2^{\pi}, ..., u_n^{\pi}$. Найчастіше зустрічаються *односортні* мови;
- Cnst – множина (можливо, порожня) *констант* мови Ω. Кожній константі з ∈ Cnst приписано певний сорт $\pi \in$ Srt;
- Fn – множина (можливо, порожня) *функціональних символів* мови Ω. У випадку односортної мови для задання виду функціонального символу досить указати кількість його аргументів;
- Pr – непорожня множина *предикатних символів* мови Ω.

Формальна аксіоматична теорія визначається набором T=<Ω, X>, де Ω – логіко-математична мова, X – множина *нелогічних аксіом* теорії T.

Питання несуперечності, повноти і можливості розв'язання формальних теорій. Вивчаючи формальну теорію, потрібно вирішувати питання її *несуперечності*, *повноти* та *можливості розв'язання.* З цими питаннями пов'язані інші. Приміром, з несуперечністю і повнотою безпосередньо пов'язані проблеми коректності теорії, незалежності системи аксіом, компактності теорії тощо.

Несуперечність теорії визначається тим, що в ній не виводиться водночас деяке твердження і його заперечення. Знаменита друга теорема Геделя, отримана в 30-і роки XX століття, стверджує, що несуперечність досить складної теорії не може бути встановлена засобами самої теорії. На сьогодні несуперечність таких теорій, як елементарна геометрія та арифметика, добре вивчена та досить надійно обґрунтована.

Великий інтерес представляє вивчення повної теорії. Теорія називається *повною*, якщо будь-яке її твердження можна довести або спростувати (тобто довести його заперечення) у межах цієї теорії. У багатьох математичних теоріях час від часу виникають конкретні проблеми, що не вдається ні довести, ні спростувати. Це буває наслідком технічних причин, і через певний час проблему все-таки вдається вирішити. Однак інколи проблему неможливо ані довести, ані спростувати в рамках теорії. Теорема Геделя про неповноту стверджує, що всяка теорія обов'язково містить твердження, які

неможливо ані довести, ані спростувати в її рамках, проте деякі важливі теорії виявляються повними. Прикладом повної теорії є елементарна геометрія.

Важливим питанням є *можливість розв'язання* теорії, тобто існування алгоритму, який дозволяє за будь-яким твердженням теорії з'ясувати, чи є воно істинним або хибним. Тарський у 1948 р. довів можливість розв'язання елементарної геометрії, у той же час доведено, що багато теорій, приміром, арифметика та теорія множин, є нерозв'язними.

Предикат P – це конструкція, що відображає певний зв'язок між об'єктами або їх властивостями. Предикат – логічна функція, що приймає значення "*істина*" або "*хибність*" залежно від значень своїх аргументів. Кількість аргументів предиката називають його арністю.

Предикат – певне твердження, істинність або хибність якого залежить від значень його аргументів. Приміром, істинність бінарного предикату "*X > 0*" залежить від значення X. Предикат приймає значення "істина", якщо X=5, або значення "хибність", якщо X = –3. Об'єкти, зв'язані предикатом *P,* називають *термами*. Терми бувають трьох типів:

- константа (позначає індивідуальний об'єкт або поняття);
- змінна (позначає в різний час різноманітні об'єкти);
- складовий терм – функція.

Константа – це терм, що посилається на одну конкретну сутність. Її можна розглядати як ім'я цієї сутності.

Змінна – терм, що посилається на якусь довільну, не визначену конкретно сутність.

Терми, побудовані за допомогою функціональних символів, виступають у ролі складних імен, які дозволяють іменувати сутності на основі їх зв'язку з іншими сутностями

Предикати можуть об'єднуватися у формули за допомогою логічних зв'язок. Логічні зв'язки – це функції, які дозволяють будувати з одних формул інші, більш складні.

Формули логіки предикатів визначаються рекурсивно:

- якщо A – предикат, тоді A – це формула;
- якщо A та B – формули, тоді об'єднання A та B через логічні зв'язки – теж формули;
- інших формул не існує.

Багато формул логіки предикатів використовують квантори *загальності* (∀ – "для всіх") та *існування* (∃ – "існує"), котрі визначають область значень змінних – аргументів предикатів.

Квантори зв'язують змінні предикатів, на які вони діють, і перетворюють предикати у висловлення.

Підформула A формули $\forall xA$ називається областю дії квантора $\forall x$. Якщо A не містить змінну x, тоді зміст формул A та $\forall xA$ однаковий.

Входження змінної x у формулу A називають *зв'язаним*, якщо x входить в область дії квантора загальності: $A = \forall xB$. У протилежному випадку входження x у формулу називають *вільним*.

Змінна називається вільною у формулі, якщо вона має вільне входження до неї. Змінна називається зв'язаною у формулі тоді й тільки тоді, коли всі її входження в цю формулу зв'язані.

Для будь-яких формул A, B і C теорії T логічні аксіоми завжди істинні. Приміром, такі формули є логічними аксіомами: $A \supset (B \supset A)$; $(A \supset (B \supset A)) \supset ((A \supset B) \supset (A \supset C))$.

Власні аксіоми змінюються від теорії до теорії і відображають закономірності конкретної ПрО, яку описує дана теорія.

Формальна теорія T визначається:

- *алфавітом*. Алфавіт вважається заданим, якщо можливо однозначно встановити належність символів алфавіту. Скінченні послідовності символів алфавіту називаються *виразами* теорії T;
- *формулами*, що є підмножиною виразів теорії T. Множина усіх формул теорії T називається *мовою* теорії T;
- *аксіомами*, що є підмножиною формул теорії T;
- *правилами виведення* $R_1,...,R_n$ – скінченною множиною відношень між формулами.

Якщо дані n формул знаходяться у відношенні R_i з формулою A, тоді A називається *безпосереднім наслідком* даних n формул за правилом R_i.

Виведення у T – така послідовність формул A_1, ..., A_n, що будь-яка формула $A_i, i = \overline{1,n}$, є аксіомою теорії T або безпосереднім наслідком інших формул за одним з правил виведення.

Формула A теорії T називається *теоремою* теорії T, якщо існує виведення в T, у якому останньою формулою є A.

Найбільше застосовують різновид формальних теорій, що має назву *теорії першого порядку*. Ці теорії дозволяють подавати судження, поділені на суб'єкт, який є предметом твердження, і предикат, що відображає те, що стверджується в судженні про суб'єкт. Крім того, теорії першого порядку можуть виражати судження з квантифікацією. Приміром, "*для всіх сутностей*

справедливо <судження>". Теорії першого порядку, які не мають власних аксіом, називаються *численням предикатів першого порядку*.

Правилами виведення у всякій теорії першого порядку є:

- MP (*modus ponens*): з A і A ⊃B випливає B;
- правило узагальнення: з A випливає $\forall x_i$ A.

При інтерпретації предикатів можливі три основні ситуації:

- формула A *здійсненна,* тобто існує набір констант *x*, що робить предикат істинним, – ∃xA ;
- формула A *загальнозначуща,* тобто вона здійсненна для будь–яких наборів – ∀xA ;
- формула *нездійсненна* (суперечлива), тобто не існує набір констант *x*, що робить предикат істинним, – ¬∃xA .

Нечіткі множини та нечітка логіка

У багатьох ПрО часто зустрічаються елементи нечіткості, що не обов'язково носять імовірнісний характер. Зокрема, нечіткість може бути лінгвістичною, тобто виникати з властивостей мови опису задач (приміром, такі характеристики, як “складний” або “простий”, не задають кількісну міру та мають сильну контекстну залежність – так, проста операційна система складніша за складну лабораторну роботу). Бажано, щоб система штучного інтелекту (ШІ) могла бути здатною працювати з такими знаннями.

Теорія нечітких множин – це етап на шляху до зближення точності класичної математики і неточності навколишнього реального світу. Основна ідея теорії нечітких множин полягає в тому, що спосіб міркувань природною мовою, властивий людині, не може бути описаний у рамках традиційних математичних формалізмів, яким притаманна однозначна інтерпретація, у той час як використання природних мов припускає багатозначну інтерпретацію. Теорія нечітких множин надає формальний апарат для строгого математичного опису розпливчастих тверджень (“fuzzy logic”), дозволяючи тим самим автоматизувати обробку наближених і нечітких суджень людини. Зараз існує кілька підходів до формалізації нечітких понять.

В нечіткій логіці належність елементу множині визначається характеристичною функцією належності, яка може приймати значення з інтервалу [0,1] (нечіткі множини Заде [289]), в скінченних або нескінченних дистрибутивних решітках (нечіткі множини Гогена). Значенням функції може бути не точка, а відрізок (Р – нечіткі множини).

Різні способи формалізації нечітких понять дозволяють описувати складні і недостатньо досліджені системи. Теорія нечітких множин корисна при створенні ІАС, що здатні взаємодіяти з користувачем й отримувати інструкції природною мовою. Саме до таких ІАС відносяться інтелектуальні програмні агенти.

Реальні дані часто виражені у вигляді кількісних показників (приміром, інформація, що надходить від різноманітних сенсорів в електронній формі). Коректно перетворити їх до вигляду, придатного для індуктивного узагальнення, можна за допомогою теорії грубих множин, ідея яких була запропонована Паулаком [174]. Неважливі відмінності між об'єктами класифікації відкидають, якщо об'єкти належать до одного класу. Кількісна оцінка певної величини заміняється на визначення інтервалу значень, у який потрапляє ця величина. Інтервал при цьому являється нечітким. Наприклад, маючи зріст людини в сантиметрах, потрібно перейти до наближеної оцінки її росту "висока-низька". Різниця між верхнім і нижнім наближенням являє собою граничну область. Нижнє наближення "висока людина" містить усіх людей, що точно можуть бути названі високими, а верхнє – усіх людей, що можливо високі, а гранична область охоплює людей, які не можуть бути точно класифіковані ні як високі, ні як низькі. Об'єкти, значення атрибутів яких потрапляють у граничну область, не включаються в навчальну вибірку, тому що не дозволяють точно класифікувати об'єкти.

1.3. Знання і моделі їх подання

Існують різні визначення поняття "*знання*". Знаннями називають формалізовану інформацію, яку використовують у процесі рішення задачі для отримання нової інформації [259]. Перехід від обробки даних до обробки знань – результат розвитку й ускладнення структур, що є об'єктом ІТ. Знання відображають закономірності певної ПрО (принципи, зв'язки, закони), отримані в результаті практичної діяльності та досвіду розв'язання її задач.

Подання знань – домовленість про те, як описувати реальний світ, вимагає їх формалізації і структурування. Знання можуть різнитися за рівнем абстракції (конкретні та абстрактні) та рівнем деталізації (повні або неповні, достовірні або недостовірні).

Властивості знань

Існують два підходи до обробки знань: логічний та евристичний. У *логічному (*формальному*)* підході основна увага

приділяється вивченню та застосуванню теоретичних методів подання знань (приміром, створення моделей подання знань на основі логічного числення першого порядку). *Евристичний (*когнітивний*)* підхід орієнтується на відображення специфіки конкретної ПрО.

Властивості знань [300]:

- *інтерпретованість* – знання мають семантичне наповнення;
- *структурованість* – знання про складні об'єкти можуть бути подані як декомпозиція знань на простіші об'єкти та зв'язки між ними;
- *зв'язність* – знання відображають відношення між фактами та явищами (причинно–наслідкові, родо–видові тощо);
- *ситуативна сумісність* знань – знання відображають ситуації, припустимі при взаємодії об’єктів певної ПрО;
- *активність* – знання забезпечують генерування нової інформації.

Знання ПрО містять опис об'єктів, їх оточення та відношення між ними [333]. Знання визначаються як “...основні закономірності ПрО, що дозволяють людині вирішувати конкретні виробничі, наукові й інші задачі, тобто факти, поняття, взаємозв'язки, оцінки, правила, евристики (*фактичні* знання), а також стратегії прийняття рішень у цій області (*стратегічні* знання)” [264].

Інтенсіональні знання – це знання про зв'язки між атрибутами (ознаками) об'єктів ПрО. Вони оперують абстрактними об'єктами, подіями і відношеннями. *Екстенсіональні знання* характеризують конкретні об'єкти, їхній стан, значення параметрів у просторі та часі.

База знань (БЗ) – це формальне подання цілісної, несуперечливої сукупності суджень, що відображають знання ПрО. Екстенсіональна частина БЗ складається з усіх явних фактів ПрО, а інтенсіональна – це сукупність правил виведення та процедур, за допомогою яких виробляють нові твердження.

Знання і дані – різні аспекти інформації. Знання – це загальна, відносно постійна і незмінна частина інформації. Дані можна розглядати як доповнення до знань, оскільки вони несуть специфічну інформацію для різних об'єктів. Це більш динамічна частина інформації. Відповідно, знання та дані взаємно доповнюють один одного. Наприклад, факт, що крива А є колом з радіусом 1 см, – це дані, а факт, що довжину кола можна обчислити за формулою $l = 2\pi r$, – це знання.

Між поняттями "дані" і "знання" не існує строгого розмежування. Те, що в даний момент є "знаннями" у процесі роботи системи, в іншому випадку може стати "даними", аналогічно і "дані" у певній ситуації можуть бути інтерпретовані як "знання". Якщо розглянути систему із загальним описом фігур, то $l = 2\pi r$ – це дані, що належать до опису кола.

Знання інтерпретуються, оскільки несуть інформацію про свій зміст. Між ними існують відношення, що визначають структуру знань: *ситуаційні* (горизонтальні) – для відображення ситуаційних і причинно-наслідкових відношень і *класифікаційні* (вертикальні) – для відтворення загальних ознак, родо-видових зв'язків і успадкування властивостей.

Традиційні ІС включають алгоритмічні знання, що містяться в програмах. Ці знання є невід'ємною частиною програм і вводяться розробниками програм заздалегідь.

Як дані, так і машинні команди завжди інтерпретуються тільки людиною. Їх не можливо розмежувати в пам'яті за допомогою формального правила. Це обумовлено третім принципом Дж. фон Неймана: "*Дані і програма мають знаходитися в пам'яті водночас*", який дозволяє розглядати програму як набір даних.

Існують різні типи знань. *Фактографічні (*декларативні) знання – це кількісні та якісні характеристики конкретних об'єктів і їх елементів. Під декларативними знаннями розуміють твердження типу "*А – це В*", приміром, "*програмний агент – це комп'ютерна програма*", "*Марія – це ім'я людини*", "*Келлі Лада де Мандрака – це стафордширський тер'єр*". Декларативні знання не містять у явному вигляді опис процедур перетворення і не залежать від того, де і коли використовуються. Моделювання ПрО в такій формі потребує повного опису всіх можливих її станів. *Процедурні знання* – це способи перетворення декларативних знань для розв'язання проблем ПрО. При процедурному поданні знань не потрібно зберігати інформацію про всі можливі стани ПрО, досить мати лише опис початкового стану та процедури, що генерують необхідні стани на основі початкового.

Знаннями ІАС називають трійку <F,R,P>:

- F – сукупність фактів ПрО;
- R – сукупність правил виведення, які дозволяють на основі відомих знань набувати нових знань;

- P – сукупність процедур, що визначають, яким чином слід застосовувати правила виведення.

Приклад.

F ={f_1="*Лада – стафоршидський тер'єр*", f_2="*Рижик – кіт*"};

R={r_1="*стафоршидські тер'єри нападають на інших соб*ак", r_2="*стафоршидські тер'єри нападають на котів*", r_3="*стафоршидські тер'єри не нападають на людей*"}.

Формально це можна записати так:

- f_1=*відноситься* (Лада, стафоршидські тер'єри);
- f_2=*відноситься* (Рижик, коти);
- r_1=*нападати* (стафоршидські тер'єри, інші собаки),
- r_2=*нападати* (стафоршидські тер'єри, коти),
- r_3=¬*нападати* (стафоршидські тер'єри, люди).

За допомогою процедур виведення, приміром, P={ЯКЩО *нападати*(x,y)∧*відноситься*(X,x)∧*відноситься*(Y,y), ТОДІ *нападати*(X,Y)}, можна отримати відповідь на питання "*Чи нападе Лада на Рижика?*". З r_2, f_1 та f_2 отримуємо новий факт – *нападати*(Лада, Рижик).

У системах ШІ значного поширення набуло символьне подання знань. *Модель подання знань* (МПЗ) – це система формалізмів (понять і правил), відповідно до яких ІС зберігає знання та здійснює операції над ними.

Методи подання даних базуються на чітких алгоритмах, а МПЗ – на уявленнях експерта. МПЗ часто буває суперечливою, неповною та нечіткою і потребує формалізації, яка здійснюється з використанням багатозначної логіки, теорії нечітких множин, імовірнісних і статистичних методів.

МПЗ необхідні для формалізації механізмів виведення нових знань з уже існуючих. Найпоширеніші МПЗ:

- логічні моделі;
- продукційні моделі;
- семантичні мережі;
- фреймові моделі.

Термін "*управління знаннями*" ("Knowledge Management") з'явився в середині 90-х років. Для створення систем управління знаннями використовують різноманітні технології, такі, як бази знань і сховища даних (Data Warehouse), інтелектуальні інформаційно-пошукові системи, експертні системи тощо.

Управління знаннями – це сукупність процесів, що пов’язані зі створенням, поширенням, обробкою і використанням знань.

Логічні моделі

Основна ідея побудови *логічних* МПЗ полягає в тому, що вся інформація, необхідна для розв’язання прикладних задач, розглядається як сукупність фактів і тверджень – формул логіки. Знання відображаються сукупністю таких формул, а одержання нових знань зводиться до реалізації процедур логічного виведення. *Логічна* модель – це сукупність фактів і тверджень, що є формулами логіки. Знання описуються сукупністю формул, а отримання нових знань зводиться до реалізації процедур логічного виведення.

В основі логічних МПЗ лежить поняття формальної теорії, що задається четвіркою: S = <B, F, A, R>, де B – скінчена множина базових символів (алфавіт); F — множина формул; A – множина аксіом; $A \subseteq F$; R – скінченна множина відношень між формулами, що називається правилами виведення.

Переваги логічних моделей подання знань полягають у тому, що у якості базису використовується класичний апарат математичної логіки, методи якого досить добре вивчені і формально обґрунтовані. Логічні МПЗ підтримують ефективні процедури виведення. У БЗ можна зберігати тільки множини аксіом, а інші знання одержують з них за правилами виведення.

Логічні моделі подання знань реалізуються засобами логіки предикатів. Приклад формальної системи S – числення предикатів першого порядку. Твердження “*X більше Y*" можна подати бінарним предикатом Q(X,Y), у якому предикатний символ Q позначає відношення ">". Предикат Q(X,Y) буде істинним, якщо для чисел X та Y виконується умова X>Y, та хибним – у протилежному випадку.

Числення предикатів із *кванторами* (логіка предикатів) є розширенням числення висловлень, у якому для виразу відношень між об'єктами ПрО можуть використовуватися пропозиції, що включають не тільки константи, але і змінні.

Продукційні моделі

Термін "*продукція"* введений Постом. Продукційна модель – найпростіший засіб подання знань. Вони складаються з продукційних правил типу "*Якщо A, тод B*". A називають посилкою (умовою, антецедентом), а B – дією (консеквентом). Це означає, що "*якщо всі умови A є істинними, тоді B – також істинне*" або "*якщо всі умови A*

виконуються, тоді потрібно виконати дію B". Приміром, якщо дати Келлі Ладі команду та показати шматок м'яса, тоді вона дасть лапу. Якщо ж м'ясо не показувати, то лапу вона не дасть, навіть почувши команду.

Змінні, які є в A і B, показують, що продукційне правило містить універсальні знання, абстраговані від конкретних значень цих змінних. Змінна, використана у виведенні і різноманітних посилках, може мати різні конкретні значення.

Простота та наочність подання знань за допомогою продукцій зумовила їх застосування в багатьох ІС. Продукційна система включає такі компоненти:

- БД з множиною фактів;
- базу правил з набором продукцій;
- інтерпретатор (механізм логічного виведення) або правила роботи з продукціями.

Продукційні системи поділяють на два типи: з прямими й оберненими виведеннями. При прямому виведенні міркування ведеться від даних до гіпотез, а при оберненому здійснюється пошук доказу або спростування деякої гіпотези. Часто використовуються комбінації прямого і оберненого ланцюжка міркувань.

Продукції порівняно з іншими МПЗ мають такі переваги:

- *модульність:* окремі продукційні правила можуть бути додані, видалені або змінені в БЗ незалежно від інших;
- *автономність:* кожне продукційне правило є самостійним елементом знань (локальне джерело знань), окремі продукційні правила пов'язані одне з одним лише через потік даних, які вони обробляють;
- *простота інтерпретації:* «прозора» структура продукційних правил полегшує їхню змістовну інтерпретацію;
- *природність:* більшість людських знань може бути записана у вигляді продукцій;
- *подібність структури:* основні компоненти продукційної системи можуть застосовуватися для побудови інтелектуальних систем з різною проблемною орієнтацією;
- *гнучкість ієрархії понять:*зміна правила спричиняє зміну в ієрархії).

Подання знань за допомогою продукцій іноді називають "пласким", тому що в продукційних системах відсутні засоби для

встановлення ієрархії правил. Обсяг БЗ продукційних систем зростає лінійно, у міру включення в неї нових фрагментів знань, у той час як у традиційних алгоритмічних системах, що використовують дерева рішень, залежність між обсягом БЗ і кількістю знань є логарифмічною.

Недоліки продукційних систем виявляються тоді, коли зростає кількість правил. Продукційні системи схожі на логічні моделі, тому на їхній основі можна організовувати ефективні процедури виведення, але вони краще (порівняно з класичними логічними моделями) відображають знання.

Семантичні мережі

Семантичні мережі – це потужніші засоби подання знань порівняно з продукційним моделями.

Мережні моделі є найбільш адекватним способом формалізації подання знань у природномовних текстах. Знання описуються сукупностями трійок (a r b), де a і b – об'єкти або поняття, a r – бінарне відношення між ними.

Формально мережні моделі подання знань можуть бути задані у вигляді $H = \langle I, C_1, ..., C_n, R \rangle$, де

- I – множина інформаційних одиниць;
- $C_1, ..., C_n, i = \overline{1, n}$ – множина типів зв'язків між елементами I;
- R – відображення між інформаційними одиницями, що входять у I, із заданого набору типів зв'язків $\{C_i\}$.

Відомо близько 200 базових відношень, решта є їхньою комбінацією [7]. Приклади базових відношень: класифікації (мати ім'я, клас – підклас, елемент – клас, частина – ціле), ознакові або атрибутивні (мати ознаку або значення ознаки), кількісні (мати міру або значення міри), порівняння (більше – менше, схожий – не схожий), часові, просторові, каузальні (причина – наслідок, бути метою, бути мотивом), інструментальні (служити для, бути інструментом, сприяти), модальні (проблематичність, необхідність, можливість) тощо.

Залежно від типів зв'язків $\{C_i\}$ розрізняють:

- *мережі класифікації* з ієрархічними відношеннями між елементами множини *I;*
- *функціональні мереж* з функціональними відношеннями, що дозволяють описувати процедури обчислення значень одних інформаційних одиниць через інші;

- *сценарії* з казуальними відношеннями, наприклад, причинно-наслідковими, а також відношеннями типу “засіб - результат”, “інструмент - дія” тощо.

Якщо в мережній моделі допускаються зв'язки різних типів, то її називають *семантичною мережею*. Вона формалізує знання у вигляді орієнтованого графа з розміченими вершинами (поняттями) і дугами (відношеннями). Це найзагальніша модель подання знань, тому що в ній є засоби реалізації всіх характерних властивостей знань: внутрішньої інтерпретації, структурованості й активності.

Переваги семантичних мереж:

- великі виразні можливості;
- наочність;
- близькість структури мережної моделі до семантичної структури фраз природною мовою;
- відповідність сучасним уявленням про організацію пам'яті людини.

Недоліки семантичних мереж:

- семантична мережа не дає чіткого уявлення про структуру відповідної ПрО, тому її важко формувати та модифікувати;
- семантичні мережі – пасивні структури, для обробки яких необхідний спеціальний апарат формального виведення і планування;
- складність виведення на семантичних мережах, особливо при наявності багатьох відношень між її елементами (проблема пошуку рішення в семантичній мережі зводиться до проблеми пошуку фрагмента мережі, що відповідає поставленій задачі).

Класифікації семантичних мереж:

- за кількістю типів відношень:
  - однорідні;
  - неоднорідні;
- за типами відношень:
  - бінарні;
  - N-арні;
- за зв'язками:
  - типу “частина - ціле” (“клас - підклас”, “елемент - множина”;
  - функціональні зв'язки (дієслова - зв'язки: є, належить, впливає, виробляє, має тощо);
  - кількісні: (більше, менше, дорівнює,…);
  - просторові: (за, під, над, близько, від, …);
  - часові: (пізніше, протягом,…);
  - логічні (АБО, І, НЕ);

- атрибутивні: (мати властивість, значення).

Приклад семантичної мережі. Твердження "*Юля дала своїй собаці Ладі – рудому стафордширу з великими очима – шматок м'яса*" може бути передане за допомогою семантичної мережі з різними типами відношень: класифікації, атрибутивними та інструментальними (рис.1.1).

Рис.1.1. Приклад семантичної мережі

Фреймові моделі

Термін "фрейм" (frame – рамка) запропонував у 1979 р. Мінський для визначення структури знань [311]. Фрейм – це такий опис деякої сутності, подальше скорочення якого призводить до втрати вмісту цієї сутності. Фрейми призначені для опису типових ситуацій або понять.

Внутрішня структура фрейму складається з множини елементів, називаних слотами**.** Слот – це структура даних або процедура. Фрейм – абстрактне подання певного стереотипу сприйняття. Фрейм у будь-який момент може бути доповнений різною інформацією, що стосується способів і наслідків використання фрейму

Формально фрейм має структуру $f\left[\left\langle v_1, g_1\right\rangle, \left\langle v_2, g_2\right\rangle, ..., \left\langle v_k, g_k\right\rangle\right]$, де

- f – ім'я фрейму;
- пара $\left\langle v_i, g_i\right\rangle, i = \overline{1,k}$, – і-й слот;
- v_i – ім'я слоту;

- g_i – його значення.

Структура фрейму подається як
(Ім'я фрейму:
(ім'я слоту1, значення_1), (ім'я слоту2, значення_2),…,
(ім'я слоту N, значення_N))
Приклад фрейму:
Студент[<прізвище, Іванов>, <курс, 3>,
<факультет, ОФФ>,
<спеціальність, економічна кібернетика>,
<ім'я, Сидір>, <по батькові, Степанович>, ...]
Фрейми поділяють на групи:

- *фрейми-структури,* що використовуються для визначення об'єктів та понять (приміром, вексель, рахунок-фактура);
- *рольові фрейми* (приміром, клієнт, менеджер);
- *фрейми-сценарії* (наприклад, збори акціонерів, покупка акцій на електронній біржі);
- *фрейми-ситуації* (зокрема, аварія).

У рольовому фреймі імена слотів – це питання, відповіді на які є значеннями слотів. Якщо у фреймі залишити тільки імена без значень слотів, то отримаємо конструкцію, що називається прототипом фрейму, фреймом-інтенсіоналом. Фрейми з конкретними значеннями слотів називаються фреймами-екземплярами.

Способи отримання слотом значення у фреймі-екземплярі:

- за замовчуванням з фрейму-зразка;
- через успадкування властивостей фрейму типу "*відноситься до*" – "*A_Kind_Of*";
- за формулою, яка вказана у слоті;
- через спеціалізовану процедуру;
- з БД;
- через діалог з користувачем.

Фрейми бувають вкладеними, тобто значенням слоту може бути інший фрейм. Це забезпечує фреймовим мовам структурованість та зв'язність, а наявність імен фреймів та імен слотів забезпечує можливість інтерпретації ПрО.

Фрейм-опис:

[<**фрукти**>, <виноград, *молдавський 10 т*>, <яблука, *джонатан 5 т* > , <вишня, *українська 1т*>].

Рольовий фрейм:

[<**перевезти**>, <що, вугілля 3000 т>, <звідки, Донбас>, <куди, Київ>, <чим, залізничним транспортом>, <коли, в серпні 2004 року>].

Основна перевага фреймів як МПЗ полягає у тому, що вони відображають основу організації пам'яті людини. Головний недолік фреймових МПЗ – відсутність механізмів виведення.

Сценарії

Особливу роль в системах подання знань виконують стереотипні знання, що описують відомі стандартні ситуації реального світу. Такі знання дозволяють відновлювати інформацію, пропущену в описі ситуації, передбачати появу нових фактів, яких можна чекати у цій ситуації, встановлювати значення походження ситуації з погляду загальнішого ситуативного контексту.

Для опису стереотипного знання використовуються різні моделі. Серед них найпоширенішими є сценарії. *Сценарієм* називається формалізований опис стандартної послідовності взаємозв'язаних фактів, що визначають типову ситуацію ПрО. Це можуть бути послідовності дій або процедур, що описують способи досягнення цілей дійових осіб сценарію (наприклад, вступ до вищого навчального закладу, купівля квартири тощо). В ІС сценарії використовуються в процедурах розуміння природномовних текстів, планування поведінки, навчання, прийняття рішень, керування змінами середовища тощо.

1.4. Рівні розуміння інформації

ІС здатна обробляти інформацію, що надходить до неї у процесі функціонування, на семантичному рівні, тобто інтерпретувати її зміст. Можна говорити про те, що ІС "розуміє" цю інформацію. Будемо вважати, що ІС розуміє текст, що надходить на її вхід, якщо вона дає відповіді, правильні з точки зору експерта, на питання, що відносяться до тексту. В існуючих ІС можна виділити п'ять основних рівнів розуміння і два рівні метарозуміння [314].

Перший рівень характеризується схемою, яка показує, що будь-які відповіді на питання система формує тільки на основі прямого змісту, введеного з тексту. Якщо, наприклад, уведено текст: "*О восьмій годині ранку Лада пішла гуляти. О дев'ятій вона повернулася додому й отримала свій сніданок, який з'їла за одну хвилину.*", тоді на першому рівні розуміння система зобов'язана вміти відповідати правильно на питання типу: "*Коли Лада пішла гуляти?*" або "*Що робила Лада після прогулянки?*".

У лінгвістичному процесорі здійснюються морфологічний, синтаксичний і семантичний аналіз тексту і питань, що відносяться до нього. На виході лінгвістичного процесора отримують внутрішнє подання тексту і питань, з якими може працювати блок виведення. Використовуючи спеціальні процедури, цей блок формує відповіді. Розуміння на першому рівні потребує від ІС певних засобів подання даних і виведення на цих даних.

На *другому* рівні додаються засоби логічного виведення, які базуються на інформації, що міститься в тексті. Ці різноманітні логіки тексту (часова, просторова, каузальна тощо), здатні породити інформацію, що не присутня у тексті явно.

Для нашого прикладу на другому рівні можливе формування правильних відповідей на питання типу: "*Що Лада робила спочатку: гуляла чи снідала?*". Тільки побудувавши часову структуру тексту, ІС зможе відповісти на подібні питання. Схема ІС, за допомогою якої може бути реалізований другий рівень розуміння, має ще одну БЗ, в якій зберігаються закономірності, що відносяться до часової структури подій, їхньої просторової організації, каузальної залежності, а логічний блок володіє всіма необхідними засобами для роботи з псевдофізичними логіками.

На *третьому* рівні до засобів другого рівня додаються правила поповнення тексту знаннями системи про середовище. Ці знання в ІС, як правило, мають логічний характер і фіксуються у вигляді сценаріїв або процедур іншого типу. На третьому рівні розуміння ІС повинна дати правильні відповіді на питання типу: "*Де була Лада о пів на дев'яту ранку?*" чи "*Звідки прийшла Лада о дев'ятій?*" Для цього треба знати, що означає процес "*перебування на прогулянці*" і, зокрема, що цей процес є безперервним.

Схема ІС, у якій реалізується розуміння третього рівня, зовні не відрізняється від схеми другого рівня, однак у її логічному блоці мають передбачатися засоби не тільки для дедуктивного виведення, але і для виведення за сценаріями.

На *четвертому* рівні замість тексту використовується розширений текст, що породжується лише за наявності двох каналів одержання інформації: через перший до ІС передається текст, а через другий – додаткова інформація, відсутня в тексті, – мультимедійна . При комунікаціях між людьми роль другого каналу, приміром, грає зір. Більш одного каналу комунікації мають, наприклад, інтелектуальні роботи, що володіють зором або іншими сенсорами для вимірювання температури, тиску, вологості тощо.

Такі системи здатні розуміти тексти, прямо пов’язані з тією ситуацією, у якій породжується текст. На нижчих рівнях розуміння не можна зрозуміти, наприклад, текст: "*Подивіться, що робить Лада! Вона не повинна нападати на кота!*". При наявності четвертого рівня розуміння ІС здатна відповідати на питання типу: "*Чому Лада не має право нападати на кота?*" чи "*Що робить Лада?*" Якщо питання, що надійшло в систему, відповідає третьому рівню, то система видає потрібну відповідь. Якщо для відповіді необхідно залучити додаткову інформацію, то внутрішнє подання тексту і питання передаються до блоку, що здійснює співвіднесення тексту з ситуацією його породження, доступною ІС через зоровий або який-небудь інший канал фіксації ситуації зовнішнього світу.

Для відповіді на *п’ятому* рівні ІС, крім тексту, використовує інформацію про джерело тексту і загальну інформацію, що зберігається в пам'яті системи, яка відноситься до комунікації (знання про організацію спілкування, цілі учасників спілкування, привила участі у спілкуванні тощо). Теорію, що відповідає п'ятому рівню, називають теорією *мовних актів*.

Будь-яка фраза не тільки позначає певне явище дійсності, але і поєднує в собі три дії: *локуцію, іллокуцію* та *перлокуцію.* Локуція – це ті дії, що здійснює той, хто говорить, для того, щоб висловити свою думку. Іллокуція – дія за допомогою мовлення: питання, спонукання (наказ або прохання) і твердження. Перлокуція – дія, якою той, хто говорить, намагається здійснити певний вплив на слухача: "здивувати", "вмовити" тощо. *Мовний акт* можна визначити як мінімальну осмислену одиницю мовної поведінки. Кожен мовний акт складається з локутивного, іллокутивного і перлокутивного.

Для четвертого і п'ятого рівнів розуміння у правила поповнення тексту входять правила виведення, що базуються на знаннях про конкретний суб’єкт спілкування. Наприклад, система може довіряти суб’єкту, який породив текст, вважаючи, що породжуваний ним текст істинний, або сумніватися у суб’єкті і коригувати текст відповідно зі своїми знаннями про цей суб’єкт.

Наприклад, на вхід системи надходить текст: "*Міла обіцяла незабаром прийти*". Якщо про Мілу в системі немає ніякої інформації, вона може звернутися до БЗ і використати для оцінки часового покажчика "незабаром" деяку нормативну інформацію. З цієї інформації можна довідатися, що з великою імовірністю "*незабаром*" не перевищує півгодини. Але в системі може міститися спеціальна інформація про Мілу – об’єкт, про який йде мова у вхідному тексті. У

цьому випадку система, одержавши потрібну інформацію із БЗ, може приготуватися, наприклад, до того, що Міла швидше за все прийде не раніше за годину.

На *першому метарівні* відбувається зміна вмісту БЗ, яка поповнюється фактами, що відомі системі і містяться в текстах, введених у систему. Різні ІС відрізняються одна від одної характером правил породження фактів зі знань: методи індуктивного виведення і розпізнавання образів, принципи ймовірностей, розмитих виведень тощо. Але у всіх випадках БЗ виявляється апріорно неповною та у таких ІС виникають труднощі з пошуком відповідей на запити. Зокрема, стає необхідним немонотонне виведення.

На *другому метарівні* відбувається породження метафоричного знання. Правила породження знань метафоричного рівня, використовувані для цих цілей, – спеціальні процедури, що базуються на виведенні за аналогією й асоціації. Відомі на даний час схеми виведення за аналогією використовують, як правило, діаграму Лейбниця, що відображає лише окремий випадок міркувань. Ще більш спрощені схеми асоціативних міркувань.

Якщо розглядати рівні і метарівні розуміння з погляду архітектури ІС, то можна спостерігати послідовне нарощування нових блоків і ускладнення реалізованих ними процедур. На першому рівні досить лінгвістичного процесора з БЗ, що відноситься тільки до самого тексту. На другому – в цьому процесорі реалізована процедура логічного виведення. На третьому – необхідна база знань. Поява нового каналу інформації характеризує четвертий рівень. Крім процедур, пов’язаних з роботою цього каналу, з’являються такі, що погоджують між собою результати роботи двох каналів та інтегрують інформацію, що надходить по кожному з них. На п’ятому рівні реалізовані різноманітні способи виведення на знаннях і даних. Тут стають важливими моделі індивідуальної і групової поведінки. На метарівнях виникають нові процедури для маніпулювання знаннями, яких немає на нижчих рівнях розуміння.

Існують і інші інтерпретації феномена розуміння. Можна, наприклад, оцінювати рівень розуміння за здатністю ІС до пояснення отриманого результату. Тут можливий не тільки рівень пояснення, на якому ІС пояснює, що вона зробила, наприклад, на основі введеного в неї тексту, і рівень аргументації, коли система обґрунтовує свій результат, показуючи, що він не суперечить тій системі знань і даних, які вона має. На відміну від пояснення обґрунтування завжди пов'язане із сумою фактів і знань, що визначають певний момент

існування системи. Текст, введений в одних ситуаціях може бути сприйнятий ІС як істинний, а в інших – як хибний. Крім пояснення й обґрунтування можлива ще одна функція, пов'язана з розумінням текстів, – виправдання, тобто відповідність виведених тверджень системі норм і цінностей, закладених в ІС.

1.5. Онтологічне подання знань

Термін "онтологія" прийшов з філософії. Онтологія – це явна специфікація концептуалізації. У найбільш загальному випадку вона являє собою угоду про спільне використання понять, що містить засоби подання предметних знань і домовленості про методи міркувань. Неформально онтологія – певний опис погляду на світ у конкретній сфері інтересів, який складається з набору термінів і правил їх використання, що обмежує їх значення в рамках конкретної ПрО. Онтології дозволяють подати поняття так, що вони стають придатними для машинної обробки.

Використання онтологій для пояснення неявного і схованого знання дозволяє перебороти проблему семантичної гетерогенності.

Існує багато досить суперечливих визначень онтології. Приміром, в [100] вона тлумачиться як явна специфікація концептуалізації, тобто абстрактного представлення ПрО, а в [224] – як спільне розуміння певної області зацікавленості.

На формальному рівні онтологія – система, що складається з наборів понять і тверджень про ці поняття, на основі яких можна будувати класи, об'єкти, відношення, функції та теорії. Онтологія, як зразок домовленості про семантику ПрО, сприяє встановленню коректних зв'язків між значеннями елементів ПрО, створюючи умови для їх спільного використання (термін "семантика" – від французького *semantique* – вперше запропонував наприкінці XIX століття французький вчений М.Бреаль для визначення науки про зміст [361]).

Онтологія – БЗ спеціального виду із семантичною інформацією певної ПрО. Компоненти, з яких складаються онтології, залежать від парадигми подання. Але практично всі моделі онтологій містять певні *концепти* (поняття, класи), *властивості* концептів (атрибути, ролі), *відношення* між концептами (залежності, функції) та додаткові *обмеження,* що визначаються аксіомами. Концептом може бути опис задачі, функції, дії, стратегії, процесу міркування тощо.

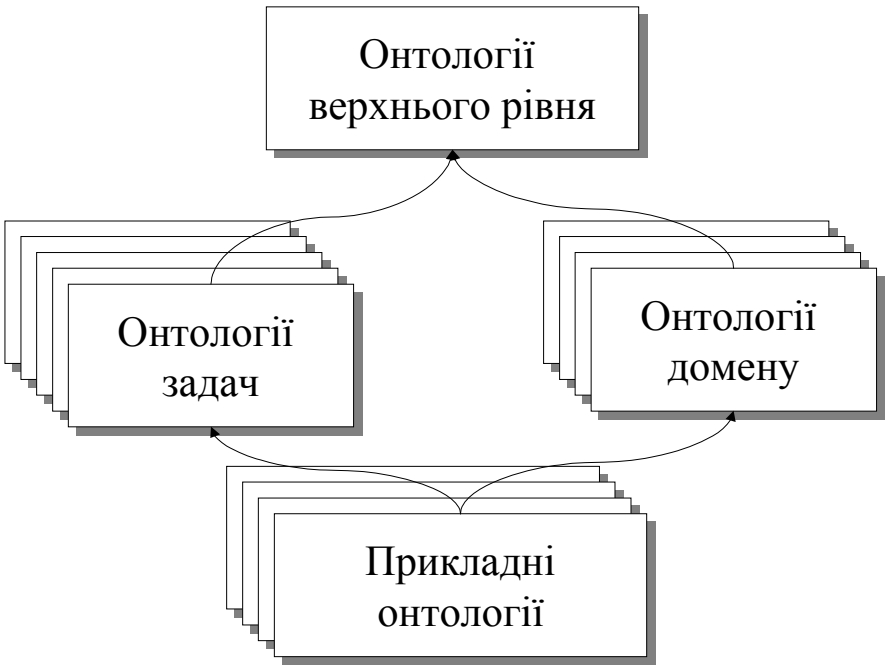


Рис.1.2. Види онтологій

Розрізняють прикладні онтології, онтології домену, онтології задачі і онтології верхнього рівня [66] (рис.1.2).

Онтології верхнього рівня описують найбільш загальні концепти, такі, як місце, час, об’єкти та події, що не залежать від конкретної проблеми або домену. Онтології домену та онтології задач відповідно описують словники, що відносяться до певної ПрО (дистанційне навчання, Інтернет-технології) або типової задачі (наприклад, діагностика, продаж тощо). Вони використовують спеціалізацію термінів, представлених в онтологіях верхнього рівня. Прикладні антології описують концепти, що залежать як від онтології задач, так і від онтології домену. Докладний аналіз онтологічного подання знань наведено в [320]. Онтологічне подання знань використовується в [252] для об’єднання ІР, що відносяться до однієї області знань, у єдиний інформаційний простір – спеціалізований Інтернет-портал знань (рис.1.3).

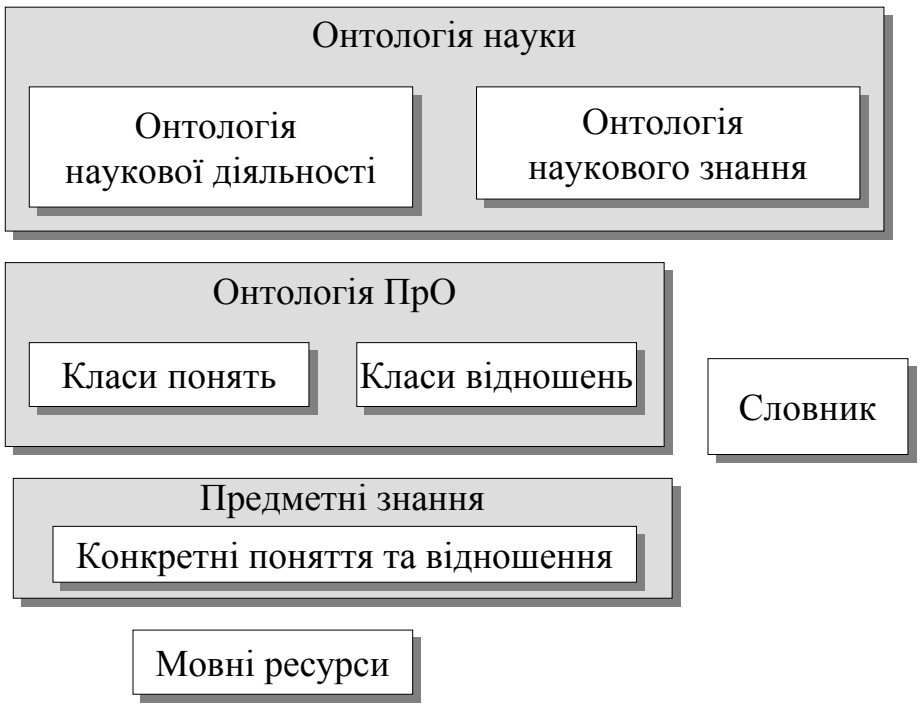


Рис.1.3. Система знань порталу

Онтологія науки містить класи понять із заданими на них семантичними відношеннями. Вона умовно розділена на онтологію наукової діяльності й онтологію наукового знання. Онтологія наукової діяльності містить загальні класи понять, що відносяться до організації наукової діяльності: вчені, організації, події, публікації тощо. Онтологія наукового знання містить метапоняття, що задають структури для опису певної ПрО, такі як розділ науки, науковий результат, метод дослідження й об'єкт дослідження, .

Онтологія ПрО відображає загальні знання ПрО, такі як ієрархія класів понять, семантичні відношення між цими класами. Предметні знання – частина знань ПрО, що містить тільки конкретні поняття і відношення.

Онтологія мови документів (словник) – це система мовних засобів, які використовуються для подання онтології ПрО. Лінгвістична інформація подається словнику за допомогою функціональних груп лексем, виділених класів понять і набору додаткових атрибутів, що відображають специфіку виразів: синоніми, омоніми тощо.

Формальна модель онтології

Формально онтологія складається з термінів, організованих у таксономію, їх визначень і атрибутів, а також пов'язаних з ними

аксіом і правил виведення [265]. *Формальна модель онтології* O – це упорядкована трійка O=<T, R, F>, де

- T – скінченна множина термінів ПрО, яку описує онтологія O;
- R – скінченна множина відношень між термінами заданої ПрО;
- F – скінченна множина функцій інтерпретації, заданих на термінах і/або відношеннях онтології O.

Відношення представляють тип взаємодії між концептами ПрО. Приклад бінарного відношення – "*є частиною*". Аксіоми використовуються для моделювання тверджень, що завжди є істинними. Моделі онтологій класифікуються таким чином:

- прості (мають лише концепти);
- на основі фреймів (мають лише концепти і властивості);
- на основі логік (наприклад, Ontolingua, DAML+OIL).

Неформальні лінгвістичні моделі часто використовуються для специфікації онтологій. У таких моделях онтологічні концепти визначені відповідно з вербальними означеннями, як у тлумачному словнику. Між концептами можуть бути встановлені певні види зв'язків. Глосарій термінів у певній ПрО, тезаурус зі своїми поняттями (концептами) і зв'язки, що визначають терміни природної мови, можуть розглядатися як онтології. Для встановлення зв'язку між вербально визначеними концептами та пошуку концептів релевантних запитів використовуються методи здобуття інформації.

Приклад. Організаційна структура всіх суб'єктів управління державними фінансами може бути подана у вигляді організаційної онтології. Онтологію, яка відображає структуру певної організації, тобто її основні компоненти та зв'язки між ними, називають *організаційною*. Основні два класи такої онтології: "*працівник*" та "*підрозділ*". Особливістю є те, що їй відповідає зв'язний граф, тобто всі елементи, що входять до складу, пов'язані з іншими її елементами. Так, формальна модель онтології Міністерства фінансів України складається з таких елементів:

- множина термінів X={"міністр", "заступник міністра", "департамент", "директор департаменту", "заступник директора департаменту", "управління", …, "IP-адреса", "контактний телефон", "e-mail", "поштова адреса"};
- множина відношень між термінами $\Re$={"підпорядковується", "є структурною одиницею", "займає посаду", "має адресу", …};
- множина функцій інтерпретації Φ={"якщо A є структурною одиницею B, а B – структурною одиницею C, тоді A є структурною

одиницею С", "якщо Х займає посаду А, тоді Х має адресу e-mail", …}.

Засоби подання та обробки онтологій

Поширення онтологічного підходу до подання знань сприяло створенню різноманітних мов подання онтологій та інструментальних засобів, призначених для їх редагування та аналізу. Онтологія в Web – це документ або файл з метаданими, який формально визначає класи, типи і властивості об'єктів, понять, термінів, а також відношення між ними, описує властивості класів і підкласів і логічні правила виведення. Приклад такого правила для поштового сервісу наводить Бернес-Лі [20]: якщо назва міста асоціюється з назвою країни, і в адресі використовується код такого міста, тоді така адреса має містити відповідний код країни. Кожному IP (сайту, БД) ставиться у відповідність (через URI типу URL) принаймні одна онтологія, що анотує цей IP. Проблеми анотування контента (змісту колекцій інформації різних типів) і сервісів визначають необхідність породження апаратом онтологій наступних типів метаданих:

- *онтології домену* (Domain ontologies) – опис (концептуалізація) важливих об'єктів, їхніх властивостей і відношень між ними (погоджений набір анотацій, понять, визначень, словників тощо);
- *онтологій задач* (Task ontologie) – описи задач і процесів, їхніх властивостей і відношень (набір характеристик фаз процесу, погоджений протокол для опису залежностей між методами);
- *онтологій якості* (Quality ontologies) – описи атрибутів знання (наприклад, анотації для поліпшення результатів, отриманих різними засобами);
- *онтологій значимості* (Value ontologies) – характеристики тих атрибутів, що відносяться до встановлення значимості, важливості контента (вартість отримання, авторитетність даних, наявність посилань на дані в інших IP);
- *персональних онтологій* (Personalization ontologies) – описи конкретних користувачів, які зверталися до відповідних IP (приміром, через посилання на персональну Web-сторінку користувача);
- *онтологій пояснень* (Argumentation ontologies) – анотації, що мають відношення до опису причин формування контенту IP (наприклад, дані будь-якого експерименту), його використання, посилань на нього.

Створення онтологій (як ручне, так і автоматизоване) потребує розробки відповідних мовних та програмних засобів, орієнтованих як на людей, так і на програмні агенти. Першими прикладами такого ПЗ є SHOE (Simple HTML Ontology Extension), XOL, OML, RDF і RDFS (Resource Description Framework Schema Language). Розробка універсальних засобів семантичної обробки інформації шляхом інтеграції усіх наявних підходів є метою проекту Semantic Web [233] консорціуму W3C (рис.1.4).

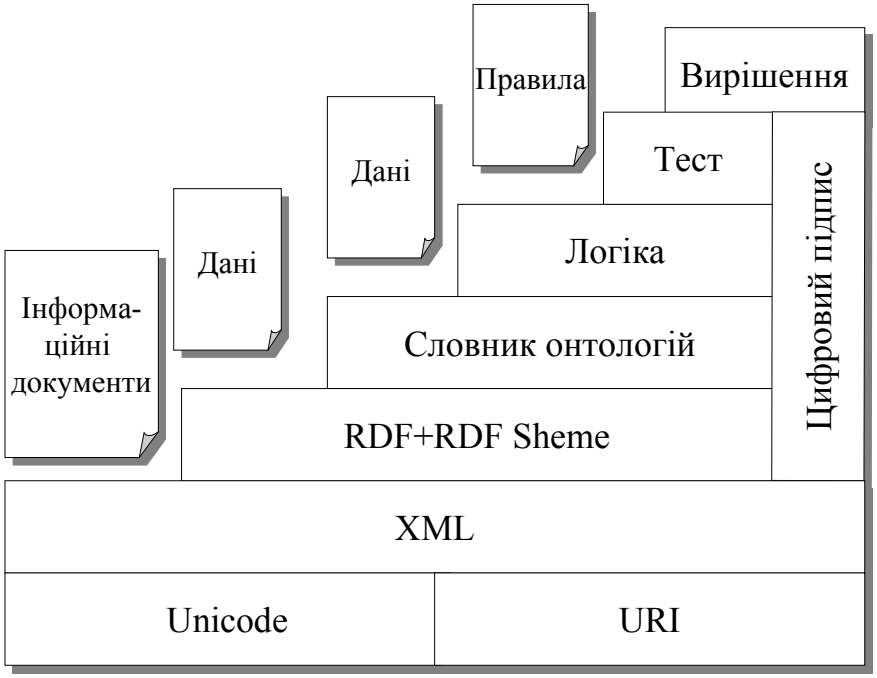


Рис.1.4. Рівні Semantic Web

У рамках проекту Semantic Web задіяні передові IT: агентно-орієнтований підхід у програмуванні – проект DAML+OIL (DARPA Agent Markup Language + The Ontology Inference Layer) [11], онтологічні системи [196], XML [48] тощо. Для формального подання онтологій розроблено мови онтологій DAML+OIL та OWL. Обидві мови засновані на RDF і RDF Schema (рис.1.5).

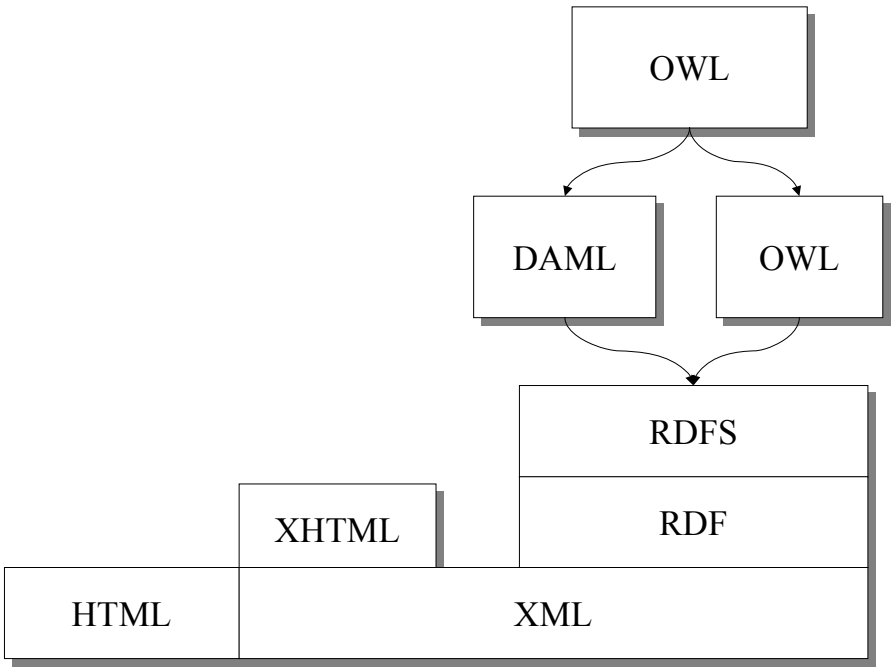


Рис.1.5. Мови Semantic Web

DAML+OIL – семантична мова розмітки Web-ресурсів, яка розширює стандарти RDF і RDF Schema більш повними примітивами моделювання. Онтологія DAML+OIL – колекція RDF–трійок. Остання версія мови DAML+OIL забезпечує багатий набір конструкцій для створення онтологій і розмітки інформації таким чином, щоб їх могла читати і розуміти машина.

Онтологія DAML+OIL включає:

- заголовки (headers);
- елементи класів (class elements);
- елементи властивостей (property elements);
- екземпляри (instances).

Мова подання онтологій OIL містить визначення властивостей (slot-def) і класів (class-def). Визначення Slot-def описують відношення між двома об'єктами. Class-def зв'язує ім'я класу з його описом і містить ряд компонентів:

- *тип визначення* defined (“обумовлений”) або primitive (“примітивний”);
- *обмеження властивостей* (slot constraint) – припустимі значення властивості для екземпляра класу;.
- *підклас* (subclass-of) пов'язує клас зі списком, що складається з одного або кількох виразів класів (class expression): імен класів, обмежень властивостей та складених з них булевих виразів довільної складності. Це підклас класів, заданих цими виразами.

Обмеження властивостей містять такі компоненти:

- ім'я (name) – рядок, що позначає властивість, на яку накладається обмеження;
- значення типу (value-type) – список з одного чи кількох виразів класів, що являють собою діапазон обмежень для властивості стосовно до обумовленого класу;
- значення (has-value) – список з одного або кількох виразів класів, у яких всі екземпляри класу, визначені за допомогою деякої властивості, мають бути пов'язані цією властивістю принаймні з одним екземпляром кожного виразу класу в списку.

Ontology Library – класифікація онтологій DAML – складається з груп онтологій, об'єднуваних наступними параметрами:

- унікальним ідентифікатором ресурсу URI;
- датою подання;
- ключовими словами;
- каталогом Open Directory Category;
- класами;
- властивостями;
- використовуваними просторами імен;
- джерелом фінансування (організацією);
- підлеглими організаціями.

Мова подання онтологій OWL [169] розширює можливості XML, RDF, RDF Schema та DAML+OIL. Ця мова базується на DAML+OIL. Проблеми, що виникли в DAML+OIL, викликані постійною зміною ядра специфікацій RDF, на якому ґрунтується DAML+OIL. OWL дозволяє усунути деякі обмеження порівняно з DAML+OIL, а також здатний прямо вказувати симетричність властивості.

Онтологія OWL (Web Ontology Language) є послідовністю аксіом і фактів, а також посилань на інші онтології. Вони також містять компонент для запису авторства та іншої подібної інформації. Онтології OWL є документами Web, на них можна посилатися через URI. Формальний опис OWL-онтології:

<ontology> ::= Ontology ([<authorship-etc>] {<directive>})

<authorship-etc> ::=...

<directive> ::= <imports> <directive> ::= <axiom>

<directive> ::= <fact> <imports> ::= imports (<URI>)

Найфундаментальніші поняття певної ПрО мають відповідати класам, що знаходяться в корені різних таксономічних дерев. Кожен екземпляр в онтології OWL належить до класу *owl:Thing*, а кожен встановлений користувачем клас автоматично є підкласом owl:Thing.

Специфічні для ПрО кореневі класи визначаються простим оголошенням іменованого класу. OWL також задає порожній клас: *owl:Nothing*. Визначення можуть бути розширюваними і розподіленими.

Фундаментальним таксономічним конструктором для класів є *rdfs:subClassOf*. Він зв'язує більш окремий клас з більш загальним. Якщо X – підклас Y, то кожен представник X – також представник Y. Відношення *rdfs:subClassOf* є транзитивним. Якщо X – підклас Y і Y – підклас Z, тоді X – підклас Z.

Визначення класу містить дві частини: назву або посилання на неї та список обмежень. Кожний вираз, який безпосередньо міститься у визначенні класу, уточнює властивості представників цього класу. Представники класу належать до перетину зазначених обмежень. Для визначення екземпляра досить оголосити його членом якогось класу.

Властивості дозволяють затверджувати загальні факти про члени класів і про екземпляри. Вони являють собою бінарні відношення. Розрізняють два типи властивостей:

- *властивості-значення* – відношення між представниками класів і RDF-літералами або типами даних, обумовлених XML Schema;
- *властивості-об'єкти* – відношення між представниками класів.

При наданні властивості існує багато способів обмежити це відношення. Можна задати домен і діапазон. Властивість може бути визначена як спеціалізація (підвластивість) існуючої властивості. Можливі і більш складні обмеження. Властивості, так само як класи, можуть бути організовані в ієрархію.

OWL використовує більшість убудованих типів XML Schema. Посилання на ці типи здійснюються за допомогою URI для http://www.w3.org/2001/XMLSchema.

Формальна семантика OWL описує, як отримати логічні наслідки, маючи таку онтологію, тобто здобути факти, що не представлені в ній безпосередньо, проте випливають з її семантики. Ці наслідки можуть бути засновані на одному документі або множині розподілених документів, які комбінуються з використанням спеціальних механізмів OWL.

Онтологія OWL відрізняється від схеми XML тим, що це є поданням знань, а не форматом повідомлень. Однією з її переваг є наявність інструментального ПЗ, призначеного для аналізу знань, поданих мовою OWL. Ці інструменти забезпечують загальну домено-незалежну підтримку онтологічного аналізу.

OWL має три діалекти, що різняться за виразністю: OWL Lite, OWL DL та OWL Full.

OWL Lite – найпростіший варіант, призначений для тих користувачів, які потребують класифікувати ієрархії і використовують прості обмеження. OWL Lite забезпечує швидку міграцію тезаурусів і інших таксономій.

OWL DL орієнтований на тих користувачів, які потребують максимальної виразності без утрати повноти обчислень та гарантованого завершення всіх обчислень у визначений час. OWL DL містить всі мовні конструкції OWL з обмеженнями поділу типу (клас не може бути приватною властивістю, а властивість – індивідом або класом). Назва OWL DL пов'язана з його відповідністю дескриптивній логіці.

OWL Full призначається для користувачів, яким потрібна максимальна виразність і синтаксична потужність RDF без обчислювальних гарантій. Наприклад, у OWL Full клас може одночасно розглядатися і як сукупність екземплярів, і як екземпляр. Інша суттєва відмінність від OWL DL у тому, що owl:DatatypeProperty може бути позначена як owl:InverseFunctionalProperty. OWL Full дозволяє такі онтології, що розширюють склад визначеного словника RDF або OWL.

OWL припускає відкритість, тобто клас спочатку може бути визначений в одній онтології, а потім розширений в інших. Внаслідок цього судження є монотонними. Нова інформація не спростовує попередню: факти і наслідки можуть тільки додаватися до онтології і не можуть видалятися. Тому інформація може бути суперечною, і розробник онтології має це враховувати.

Рівень аналізу знань, що отримав назву онтологічного аналізу, запропонований в [9]. В основі цього підходу є опис системи в термінах сутностей, відношень між ними і перетворення сутностей, що виконується в процесі рішення певної задачі.

Для структурування знань ПрО використовуються три основні категорії онтологій:

- *статична,* до якої входять сутності ПрО, їх властивості і відношення;
- *динамічна,* що визначає ситуації, які виникають у процесі рішення проблеми, і спосіб перетворення одних ситуацій в інші;
- *епістемічна,* що описує знання, які керують процесом переходу від однієї ситуації до іншої.

У цій схемі є певна відповідність з рівнями концептуалізації знань і епістеміологічного аналізу в структурі, що запропонована в [238]. Проте на нижніх рівнях логічного аналізу й аналізу впровадження така відповідність уже не проглядається. Ця схема онтологічного аналізу досить абстрактна, але вона дозволяє спростити аналіз слабоструктурованих задач.

Онтологічний аналіз припускає, що розв'язувана проблема, може бути зведена до проблеми пошуку, але при цьому не розглядається, яким саме способом потрібно виконувати пошук.

На сьогодні існує цілий ряд інструментів для онтологічного аналізу, які, крім редагування і візуалізації, підтримують документування, імпорт та експорт онтологій різних форматів і мов подання, їх об’єднання, порівняння тощо. Детальний аналіз цих інструментів наведено в [319].

Система *Ontolingua* [70] є першим інструментом інженерії онтологій. Редактор онтологій – найважливіше застосування сервера Ontolingua. Крім редактора онтологій Ontolingua містить мережний застосунок Webster, призначений для визначення концептів, сервер, який забезпечує доступ до онтологій Ontolingua за протоколом обміну знаннями OKBC (Open Knowledge Base Connectivity), та Chimaera – інструментарій для аналізу та об'єднання онтологій.

Protégé [151] – локальна, вільно розповсюджувана Java-програма, призначена для побудови (створення, редагування і перегляду) онтологій ПрО. Protégé включає редактор онтологій, що дозволяє проектувати онтології, розгортаючи ієрархічну структуру абстрактних та конкретних класів і слотів. На основі сформованої онтології Protégé дозволяє генерувати форми отримання знань для введення екземплярів класів і підкласів.

Інструмент має зручний графічний інтерфейс. Він підтримує використання мови OWL та дозволяє генерувати html-документи, які відображають структуру онтологій (рис.1.6).

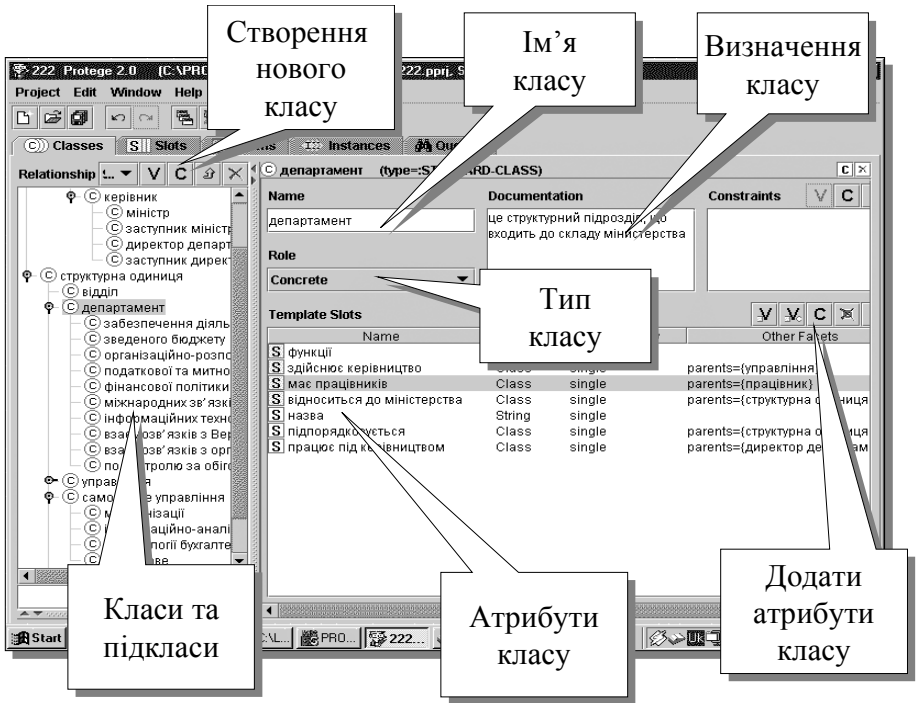


Рис.1.6. Організаційна онтологія Міністерства фінансів України у Protégé

Protégé використовує фреймову модель подання знання OKBC (Open Knowledge Base Connectivity) [163], що дозволяє адаптувати його для редагування моделей ПрО, поданих не в OWL, а в інших форматах (UML, XML, SHOE, DAML+OIL, RDF і RDFS тощо).

DOE **[**49**] –** простий редактор онтології, який надає можливість споживачеві створювати онтології. Процес специфікації складається з трьох етапів. На першому етапі користувач будує таксономію понять і відношень, явно окреслюючи позицію кожного елемента (поняття) в ієрархії. Для кожного поняття визначається, що відповідає чотирьом принципам теорії диференційної семантики. Потім користувач вказує, в чому специфіка поняття відносно його батька та у чому це поняття подібне або відмінне від його братів. Користувач може також додати синоніми та енциклопедичне визначення кількома мовами для всіх понять. На другому етапі дві таксономії розглядаються з різних точок зору. Користувач може розширити їх новими об'єктами або додати обмеження на області відношень. На третьому етапі онтологія може бути перекладена на мову подання знань, яка дозволяє використовувати це у відповідній IAC або імпортувати її в інший інструмент.

OntoEdit [165] – інструментальний засіб, що забезпечує перегляд, перевірку та модифікацію онтологій. Він підтримує такі мови подання онтологій, як OIL, RDFS, а також внутрішню мову подання знань OXML , засновану на XML. Так само як і Protege,

OntoEdit – автономний Java-застосунок, який можна локально установити на комп'ютері, але його коди закриті. Вільно розповсюджувана OntoEdit Free обмежена 50 концептами, 50 відношеннями і 50 екземплярами.

OilEd [19] – автономний графічний редактор онтологій, розроблений в рамках проекту On-To-Knowledge. Інструмент використовує OIL для подання онтологій. В OilEd відсутня підтримка екземплярів. Цей редактор застосовується як для навчання, так і для дослідження. Інструмент вільно поширюється за загальнодоступною ліцензією GPL.

WebOnto [50] розроблений для перегляду, створення і редагування онтологій. Для моделювання онтологій WebOnto використовує мову OCML (Operational Conceptual Modeling Language). У WebOnto користувач може створювати структури, у тому числі класи із множинним спадкуванням. Інструмент має ряд корисних особливостей: перегляд відношень, класів, правил, спільну роботу кількох користувачів над онтологією.

ODE (Ontological Design Environment) [72] взаємодіє з користувачами на концептуальному рівні, забезпечує користувачів набором таблиць для заповнення (концептів, атрибутів, відношень) і автоматично генерує для них код у LOOM, Ontolingua і FLogic. Інструмент одержав свій подальший розвиток у WebODE, що інтегрує всі сервіси ODE в одну архітектуру, зберігає свої онтології в реляційні БД.

Складніші інструментальні засоби потрібні для того, щоб не тільки вводити та редагувати онтологічну інформацію, але й аналізувати її, виконуючи типові операції над онтологіями.

Вирівнювання (alignment) онтологій – операція, яка дозволяє встановити різні види відповідності між двома онтологіями для того, щоб кожна з них могла використовувати інформацію іншої.

Відображення (mapping) онтології – знаходження семантичних зв'язків між подібними елементами різних онтологій.

Об'єднання (merging) онтологій – операція, яка за двома онтологіями генерує третю, що поєднує інформацію з перших двох.

Призначення інструментальних засобів обробки онтологій класифікується в [159] таким чином:

- об'єднання двох онтологій для створення нової;
- визначення функції перетворення однієї онтології в іншу;
- встановлення відображення між концептами онтологій зі

знаходженням пар відповідних концептів;

- знаходження правил відображення для зв'язку релевантних частин вихідних онтологій.

PROMPT [159] слугує для об'єднання й угруповання онтологій. Це доповнення до системи Protégé, реалізоване у вигляді плагіна. За двома онтологіями, які треба об’єднати, PROMPT будує список операцій (приміром, об'єднання термінів чи їх копіювання у нову онтологію) та передає його користувачеві, який може виконати одну з пропонованих операцій. Потім список операцій модифікується і створюється список конфліктів і можливих їх рішень. Це повторюється доти, поки не буде готова нова онтологія.

Chimaera – інтерактивний інструмент для об'єднання онтологій, що базується на редакторі онтологій Ontolingua. Chimaera дозволяє користувачеві поєднувати онтології, розроблені в різних формалізмах. Єдине таксономічне відношення, що розглядає Chimaera – відношення "підклас – суперклас". Через це Chimaera не знаходить багато відповідностей, які відшукує PROMPT.

У *OntoMerge* вихідні онтології транслюються у подання спеціальною мовою. Потім інженер онтології визначає аксіоми з'єднання, що містять терміни з обох онтологій. OntoMerge надає інструменти для трансляції екземплярів в об'єднану онтологію.

OntoMorph визначає набір операторів перетворення, які можна застосувати до онтології. Експерт задає початковий список пар і онтологій для визначення набору операторів, що мають застосовуватися до вихідних онтологій для усунення розходжень між ними.

OBSERVER об’єднує онтології з інформацією про відображення між ними та знаходячи синоніми у вихідних онтологіях.

ONION базується на алгебрі онтології. Вона надає інструменти для визначення правил артикуляції (з'єднання) між онтологіями, які враховують тільки релевантні частини вихідних онтологій. ONION використовує як лексичні методи, так і методи на основі графів. Метод встановлення лексичної подібності між іменами концептів використовує словники і методи семантичної індексації, які враховують місцезнаходження групи слів у тексті.

Інструменти OntoEdit, WebODE і KADS22 підтримують методології побудови онтологій On-To-Knowledge, ETHONTOLOGY і CommonKADS відповідно, що не заважає їм застосовуватися в інших методологіях або взагалі без них.

Більш ранні інструменти Ontolingua, OntoSaurus і WebOnto мають клієнт-серверну архітектуру. В Protégé, OntoEdit і OilEd

трирівнева архітектура, де існує чіткий поділ між збереженням онтологій, модулями бізнес-логіки застосунків і застосунків інтерфейсу користувача. Ці інструменти мають великі можливості щодо нарощування. Більшість з них зберігає свої онтології в текстових файлах, що обмежує розмір онтологій. Тільки Protege і WebODE можуть зберігати свої онтології в БД і в такий спосіб керувати великими онтологіями.

Для подання онтологій більшість інструментів комбінує фрейми і логіку першого порядку. Тільки OilEd і OntoSaurus базуються на дескриптивній логіці (DL).

Усі редактори онтологій, за винятком OilEd, Ontolingua і OntoSaurus, забезпечують графічні засоби редагування і перегляду онтологій, де класи звичайно подано вузлами на графах, а відношення – дугами між ними. Додатково до цих графічних функцій OilEd, OntoEdit Professional, Protégé і WebODE надають деяку підтримку в написанні формальних аксіом і складних виразів.

OntoEdit, Ontolingua, OntoSaurus, WebODE і WebOnto підтримують спільну розробку онтологій, надаючи окремим користувачам або їх групам дозвіл на доступ і написання різних наборів онтологій. Властивості найпоширеніших інструментів (таб.1.1) наведено в [55].

Таблиця 1.1

Редактори онтологій

Параметри	Protégé (плагін OWL)	Ontolingua	DUET	OntoEdit	RDFEdit	OilED
Мови подання онтологій	OWL	OWL	DAML+ OIL	DAML+ OIL	DAML+ OIL	RDF
Підтримка у Web	Задовільно	Задовільно	Задовільно	Добра	Задовільно	Добре
Підтримка інсталяції	Добре	Добре	Задовільно	Задовільно	Добре	Добре
Документація	Задовільно	Задовільно	Задовільно	Добре	Задовільно	Добре
Відкритість коду	Так	Ні	Ні	Ні	Ні	Так

Методологія створення онтологій

Основною характерною рисою онтологічного аналізу є поділ реального світу на складові і класи об'єктів, визначення їх онтологій – сукупності фундаментальних властивостей, що окреслюють їх поведінку та розвиток [261]. Створення онтологій – складний процес. Його успіх значною мірою залежить від вибору відповідної методології та її додержання під час роботи.

Для створення онтології потрібно виконати такі кроки:

- спланувати, для яких цілей будуть використовуватися представлені в онтології знання;
- спроектувати прототип онтології;
- розширити онтологію новими класами;
- заповнити класи онтології конкретними екземплярами;
- протестувати онтологію у взаємодії з кінцевими користувачами.

Побудова і властивості будь-якої системи можуть бути ефективно досліджені і задокументовані за допомогою словника термінів, однозначних визначень термінів цього словника, логічних взаємозв'язків між ними. Все це дозволяє втілити стандарт IDEF5 [309], що належить до сімейства IDEF.

Сімейство методологій IDEF [83] призначено для моделювання складних систем. За допомогою цих методологій можна ефективно аналізувати і відображати моделі діяльності великого спектра систем у різних розрізах. IDEF є державним стандартом США. Воно створено в рамках програми комп'ютеризації промисловості ICAM для аналізу процесів взаємодії у виробничих системах.

Процес побудови онтології, відповідно до методології IDEF5, складається з п'яти основних дій:

1) *вивчення і систематизація початкових умов.* Встановлюються основні цілі і контекст розробки онтології, а також розподіляються ролі між членами проекту;

2) з*бір і накопичення даних.* Здійснюється збір необхідних початкових даних для побудови онтології, визначаються засоби їх збереження;

3) *аналіз даних.* Ця стадія полягає в угрупованні зібраних даних і призначена для полегшення вироблення термінології;

4) *початковий розвиток онтології.* На основі відібраних даних формується попередня онтологія;

5) *уточнення та затвердження онтології.*

Для підтримки процесу побудови онтологій у IDEF5 існують спеціальні онтологічні мови: схематична мова SL (Schematic Language) і мова уточнення EL (Elaboration Language).

SL дозволяє будувати різноманітні типи діаграм і схем у IDEF5. Основна мета цих діаграм – наочно і візуально представляти онтологічну інформацію.

Семантика і позначення схематичної мови SL суттєво відрізняються від семантики і позначень інших графічних мов – частина елементів графічної схеми SL може бути змінена чи зовсім не прийматися до уваги мовою EL, тому що основною метою використання SL є створення лише допоміжної конструкції онтології, а графічні елементи SL не несуть достатньої інформації для повного подання й аналізу системи. Саме тому вони не призначені для збереження на кінцевому етапі проекту.

Метою застосування мови EL є ретельний аналіз та забезпечення повноти подання структури даних, отриманих у результаті онтологічних досліджень.

Системи подання знань, що базуються на онтологіях

Зараз велика кількість застосунків базується на поданні знань у вигляді онтологій. Розглянемо кілька широко відомих прикладів.

Метою міжнародного проекту *c* є інтелектуальний пошук у середовищі Інтернет і автоматичне накопичення нових знань. У рамках ініціативи $(KA)^2$ виділяються три основних напрямки досліджень:

- онтологічний інжиніринг (ontological engineering);
- анотація Web-сторінок;
- запити до Web-сторінок і виведення відповідей на базі онтологічних знань.

Для специфікації онтології розроблена спеціальна мова подання знань. Підмножина цієї мови служить і для формулювання запитів, а мова анотування – для «збагачення» Web-документів онтологічною інформацією. Для правильного формулювання запиту необхідно знати, які концепти присутні в онтології і які атрибути мають концепти. Тому система представлення онтологічних знань надає своїм користувачам засоби візуалізації онтології і навігації по онтології.

Проект *SHOE (Simple HTML Ontology Extensions)* орієнтований на рішення проблеми додавання до Web-сторінок семантичної інформації і співвіднесення її з онтологіями відповідних предметних

галузей. Передбачається, що, використовуючи цю інформацію, пошукові системи зможуть забезпечувати більш релевантні відповіді на запити. Зараз проектом SHOE підтримуються такі напрямки досліджень:

- розробка множин повторно використовуваних онтологій (reusable ontologies) для концептів, що найчастіше зустрічаються у Web-ресурсах;
- створення засобів проектування онтології – анотаторів знань.

1.6. Засоби здобуття знань

У багатьох випадках ПрО не має опису у вигляді правил, закономірностей тощо, які сформульовані експертами явно. Однак ця інформація присутня в неявній формі в тих рішеннях, які вони приймають в різних ситуаціях. Описи цих ситуацій разом з прийнятими експертом рішеннями являють собою набір навчальних прикладів.

Методи автоматизованого навчання (АН) можна класифікувати за трьома основними параметрами:

- стратегією навчання;
- способами подання знань;
- сферами застосування.

Існують такі стратегії навчання для здобуття знань:

- через дедуктивні міркування;
- за аналогією;
- за індукцією.

Знання, отримані за методами АН, можуть бути подані через:

- коефіцієнти математичних виразів;
- дерева;
- продукційні правила;
- логічні вирази;
- семантичні мережі;
- фрейми і схеми;
- таксономії;
- програми і процедури.

При різноманітті методів подання знань основною мовою опису даних і знань при розробці систем АН для здобуття знань є мова, що використовує множину атрибутів різного типу і відношення між ними. Багато методів АН обробляють приклади, що подаються як

вектори атрибутів. Значення атрибутів можуть бути номінальними, лінійними або мати структуру дерева.

Форма подання нових знань залежить від сфери застосування, вимог до БЗ, способів використання знань. Основними формами подання знань є продукційні правила, дерева рішень і речення логіки предикатів першого порядку.

Методи, що ґрунтуються на узагальненні, використовують способи міркування "від часткового до загального". Найчастіше цей спосіб міркування застосовується для задач виведення понять на прикладах або знаходження концептуальних описів прикладів.

Класифікація міркувань

Міркування – один із найважливіших видів розумової діяльності людини, завдяки якому вона формулює на основі певних пропозицій, висловлювань, суджень нові пропозиції, висловлювання, судження. Механізм міркування людини залишається поки що недостатньо дослідженим. Людським міркуванням властиві неформальність, нечіткість, нелогічність, широке використання образів, емоцій і почуттів, що робить надзвичайно складним їх дослідження і моделювання.

Найпростішим видом міркувань є отримання з одного або кількох висловлень нового висловлення. Міркування поділяють на необхідні (з істинності посилок A_1, A_2,..., A_n завжди випливає істинність висновку B) та ймовірні (з істинності посилок A_1, A_2,..., A_n завжди випливає ймовірність істинності одного з висновків B_1, B_2,..., B_m) [251].

Похідні висловлення – *посилки,* а нове висловлення B – *висновок* (наслідок). Можливість виведення висновку з посилок забезпечується логічним зв'язком між ними. Перевірити слушність виведення з посилок можна логічними засобами, не звертаючись до безпосереднього досвіду.

Істинні з погляду логіки висновки будуються за допомогою правил логічного виведення.

Логічне виведення – це багатоетапний процес переходу від посилок до висновку.

Залежно від наявності проміжних кроків міркування поділяються на *безпосередні* й *опосередковані*.

Нові знання можуть бути отримані різними способами. Можна вивести їх як логічний наслідок з уже існуючих знань, вивести загальне правило з наявних фактів або перенести факти та знання,

істинні для одних об’єктів, на інші об’єкти на підставі їх подібності. З цієї точки зору міркування поділяють на

- *дедуктивні* (від загального до часткового);
- *індуктивні* (від часткового до загального);
- *за аналогією* (від часткового до часткового).

Якщо процес міркування йде від одиничних або окремих фактів до загального правила, що поширюється на ці одиничні або окремі факти, то має місце *індуктивний* умовивід.

У *дедуктивному* умовиводі процес міркування йде від загального правила до знання про одиничний, окремий або менш загальний факт, на який поширюється загальне правило.

На відміну від індукції і дедукції в *традукції* (умовиводі за аналогією) посилка і висновок є судженнями однакового рівня узагальнення, тобто процес виведення йде від знання певного ступеня узагальнення до нового знання того ж ступеня узагальнення.

Дедуктивні міркування мають найбільшу доказовість, а міркування за аналогією – найменшу.

Логіка в загальному значенні – наука про закони мислення, вивчає побудову міркувань, що гарантують отримання правильних висновків з істинних посилок. Вона надає засоби точного формулювання думок, що підвищує ефективність будь-якої інтелектуальної діяльності.

Кожне поняття має зміст і обсяг. *Зміст* – комплекс ознак предмета в цьому понятті, а обсяг – сукупність об'єктів, що входять у це поняття. *Обсяг поняття* – клас (множина) об’єктів, які виділяються та узагальнюються в понятті і кожному з них притаманні ознаки, що складають зміст поняття. *Одиничне поняття* – поняття, обсяг якого складається з одного елемента.

Поняття об’єднує знання системи про клас об’єктів, властивостей, зв'язків, станів, дій, процесів або ситуацій і є найважливішим інструментом формування знань і розв’язання задач. На основі наявного запасу понять здійснюється розпізнавання, узагальнення, структуризація інформації. При розв’язуванні творчих задач виконуються різні операції підбору і перетворення понять.

У множині процесів, що включають обробку понять, виділяють два: розпізнавання і генерацію моделей елементів світу, в якому оперує система. Процеси розпізнавання стали об'єктом дослідження і автоматизації, тоді як генерація моделей є проблемою сьогоднішніх досліджень. Генерація моделей є процесом формування моделей конкретних елементів світу шляхом введення в поняття інших понять

або констант. Генерацію моделей можна розглядати як процес, обернений формуванню понять: поняття формуються шляхом узагальнення частинних моделей, моделі конкретних елементів світу будуються в результаті конкретизації понять.

Відтак, поняття – фрагмент знань, що є узагальненою моделлю класу елементів світу, в якому оперує система, що є достатньою для виконання на її основі операцій розпізнавання і генерації моделей конкретних елементів світу.

Основними типами визначення понять є:

- *порівняння* – встановлення подібності або відмінності між поняттями;
- *аналіз* – поділ цілого на складові частини;
- *синтез* – об'єднання цілого із складових частин (ознак, властивостей, відношень тощо);
- *абстрагування* – виділення в понятті певних ознак при відволіканні від інших;
- *узагальнення* – об'єднання різноманітних об'єктів в однорідні групи на підставі загальних ознак;
- *індукція* – правило породження з окремих прикладів нових об'єктів.

Розрізняють реальне і номінальне визначення поняття. В останньому випадку йдеться про вибір *імені* ("будемо називати..."). Відомо кілька способів його визначення. Найпоширеніший з них – визначення поняття через найближчий рід і видову відмінність [1].

Усі поняття поділяються на ряд класів:

- залежно від відображення виду або роду об'єктів – на *видові* й *родові*;
- залежно від кількості об'єктів відображення – на *одиничні* й *загальні*;
- залежно від відображення об'єктів або властивості, що абстрагується від об'єктів, – на *конкретні* й *абстрактні*.

Один із способів, широко застосовуваний в ШІ для визначення понять, базується на ідеї інтенсіоналу (терміни "інтенсіонал" та "екстенсіонал" походять з *семіотики* – науки про знакові системи) [17].

Інтенсіональне визначення – визначення поняття через загальніше поняття (більш високого рівня абстракції) з наведенням специфічних видових властивостей.

Екстенсіональне визначення – визначення поняття через поняття нижчого рівня ієрархії або через факти та дані, що належать до цього поняття.

Іншими словами, *інтенсіонал* – це ті загальні поняття і відношення, що характеризують множину об’єктів, предметів, явищ, а *екстенсіонал* – конкретні характеристики кожного елемента цієї множини понять і відношень.

При побудові нових понять на основі вже відомих використовують логічні операції:

- *узагальнення поняття* – логічна операція, що полягає в знаходженні для будь-якого поняття більшого за обсягом поняття, в обсяг якого входить і обсяг поняття, що досліджується;
- *категорія* – найбільш загальне поняття, для якого не існує роду;
- *обмеження поняття* – логічна операція, протилежна операції узагальнення, що полягає в знаходженні для будь-якого поняття меншого за обсягом поняття, яке входить до досліджуваного обсягу.

Операція, у результаті якої розкривається обсяг поняття, називається *розподілом обсягу поняття*. При операції розподілу необхідно виконувати такі правила:

- розподіл має бути пропорційним;
- члени розподілу мають виключати один одного;
- розподіл повинен мати одну основу;
- розподіл має бути неперервним.

Особливим випадком операції розподілу є класифікація, що становить собою деяку сукупність розподілів.

Класифікація об’єктів – процедура групування об’єктів за класами відповідно до ознак, які властиві цим об’єктам та вирізняють їх від об’єктів інших класів, при цьому кожен клас може поділятися на підкласи.

Класи об’єктів можуть мати відношення типу *вид* і *рід*, *частина* і *ціле*. У логічному співвідношенні одне з одним перебувають тільки поняття, які можна порівняти, тобто ті, у змісті яких присутні загальні ознаки.

Поняття – це матеріал для побудови *висловлень*. Елементарне висловлення складається із суб'єкта А (логічний підмет – те, про що йдеться у висловленні), предиката Р (логічний присудок – те, що затверджується або заперечується у висловленні про суб'єкт) і

квантора (“усі” – $\forall$ або “існує” – $\exists$). Логічний зв'язок між суб'єктом і предикатом висловлення – це зв’язок “є” або “не є”, який у реченні може міститися неявно. Відповідність або невідповідність цього зв'язку реальності визначає істинність або хибність судження. Приклад істинного висловлювання: “Штрих-код використовують для маркування товарів”, хибного: “Усі ІС використовують одну ОС”.

Дедуктивні міркування

Дедуктивне виведення відіграє значну роль у мисленні людини. Дедукція – це виявлення наслідків, що неявно містяться в наявній інформації. У традиційній логіці дедуктивним виведенням називають виведення від загального до часткового. У широкому розумінні дедуктивними виведеннями називають міркування, що задовольняють умові достовірності: з істинних посилок завжди отримують лише істинні висновки.

Основи теорії дедуктивних міркувань заклав ще Аристотель понад дві тисячі років тому, побудувавши першу модель міркувань, яка базується на силогізмах. Силогізм – це міркування, яке складається з двох посилок і одного висновку.

Приклад: посилки – "Келлі Лада їсть все, крім бананів"; "Кіт Рижик не є бананом", висновок – "Лада може з’їсти Рижика (але не зобов’язана це робити)".

Основні закони мислення:

• *тотожності;*

• *протиріччя;*

• *виключення третього* (у класичній логіці*);*

• *достатньої підстави*.

Відповідно до *закону тотожності*, обсяг і зміст будь-якого поняття мають залишатися незмінними впродовж усього процесу міркування. За *законом протиріччя* два суперечні висловлення не можуть бути істинними водночас, принаймні одне з них є хибним – $A \vee \neg A$. Закон *достатньої підстави* вимагає, аби жодне твердження не визнавалося справедливим без достатньої підстави (яку саме підставу вважати достатньою – це окреме питання).

У дедуктивних міркуваннях:

1) вихідні посилки міркування є істинними;

2) правильне використання істинних посилок породжує тільки істинні висновки.

У логіці особливу роль відіграють логічні форми, істинність яких справедлива внаслідок їхньої структури. Їх називають

тавтологіями, або загальнозначущими формами (приміром, А=А). Правила виведення в дедуктивних міркуваннях описують тавтології. Розглядаються два види формальних логічних конструкцій: *терми* – аналоги іменників, і *формули* – аналоги речень. Логічні зв'язки (оператори) позначають логічні операції, за допомогою яких із виразів формують більш складні судження.

Складні висловлення залежно від зв'язки поділяють на

- *сполучні*: ТА;
- *роздільні*: АБО у двох варіантах:

1) нестрога диз'юнкція ("АБО А, АБО В, АБО А та В");

2) строга диз'юнкція – ("АБО А, АБО В") (допускається тільки одна з альтернатив);

- *умовні* ("ЯКЩО А, ТОДІ В" – імплікація);
- судження *еквівалентності* ("ЯКЩО А, ТОДІ В, ЯКЩО В, ТОДІ А");
- *змішані* висловлення.

Механізм дедуктивного виведення використовується людиною відносно рідко: майже всі знання за межами математики і доказової логіки складаються з припущень, підкріплених правдоподібними міркуваннями. Проте дедуктивна логіка вивчена значно повніше порівняно з проблемою правдоподібних міркувань.

Індуктивні міркування

Індукція – метод переходу від набору спостережень до загальної закономірності, яка відповідає всім цим спостереженням.

Індуктивне виведення – виведення з наявних даних (спостережень, фактів) загальної закономірності (правила), що їх пояснює. Отримана гіпотеза використовується для пояснення наявних даних і класифікації та прогнозування нових даних.

Розрізняють *повну* (математичну) та *неповну* (емпіричну) індукцію. Індуктивні міркування від часткового до загального відображають природний шлях пізнання навколишнього світу: загальні твердження виникають внаслідок узагальнення отриманої з досвіду сукупності одиничних фактів. Загальне твердження є хибним, якщо його спростовує хоча б один факт.

Повна індукція – це формальне доведення істинності певної закономірності шляхом її перевірки для скінченної кількості фактів та обґрунтування того, що додання довільного нового факту не призводить до зміни цієї закономірності.

Повна індукція дозволяє отримати істинні знання за умови обмеженості класу об'єктів, який слід проаналізувати. Повну індукцію найчастіше застосовують у математиці для доведення теорем. Метод математичної індукції відносно натуральних чисел – це метод доведення істинності певної гіпотези $\Phi(n)$, що полягає у перевірці його істинності для $n=1, n=2,..., n=k$, та доведення того, що $\forall m$ з істинності $\Phi(m)$ випливає істинність твердження $\Phi(m+1)$.

Неповна індукція полягає в знаходженні закономірностей, яким задовольняє скінченна множина отриманих на поточний час фактів. Приміром, покуштувавши потроху різних страв, якими вас пригощають, усі з яких виявилися смачними, можна зробити висновок, що господиня добре готує.

При цьому знайдені закономірності можуть надалі спростовуватися новими спостереженнями. Приміром, якщо господиня добре готує салати, вона може не вміти пекти торти. Хибні висновки можуть виникати внаслідок використання другорядних ознак замість істотних та поспішного узагальнення. Основне призначення неповної індукції – генерація гіпотез, що можуть потім доводитися або спростовуватися іншими методами.

Приміром, узагальнивши три приклади (1<10, 2<10, 3<10), отримуємо, що всі числа менше десяти ($\forall n < 10$), але наступний приклад n=11 суперечить цій гіпотезі, тому що 11>10.

Незалежно від вигляду індуктивної гіпотези та алгоритму її формування на основі вибірки для навчання завжди існує таке розширення цієї вибірки, на якій сформована гіпотеза є хибною.

Індуктивна логіка – формальна система, що описує правила формування загальних тверджень на основі скінченної множини окремих тверджень.

Інтелектуальність поведінки будь-якої системи значною мірою визначається її здатністю навчатися за власним досвідом, тобто індуктивно узагальнювати відомі системі факти у деякі загальні правила.

Індуктивне узагальнення полягає в тому, щоб за набором прикладів (приклад – пара (x,f(x)), де x – вхідні дані, а f(x) – значення функції для x) побудувати для функції f функцію-гіпотезу h, що апроксимує f. При виборі засобів подання для такої функції виникає конфлікт між виразністю й ефективністю. Функції можна подати через логічні висловлення, многочлени, нейронні мережі тощо.

Найпростішою з таких форм подання є дерева рішень, що можуть використовуватися для широкого спектра задач класифікації.

Задачу індуктивного узагальнення можна визначити так: ІС – пара S = (U, A), де U – непорожня скінченна множина (універсум); A – непорожня скінченна множина атрибутів, з кожним із яких пов'язана фіксована скінченна множина можливих значень цього атрибута V_a, така, що для $\forall a, a \in A, \exists V_a$. Елементи U називаються об'єктами та інтерпретуються як події, стани, процеси тощо. Всім атрибутам кожного об'єкта універсуму присвоюються значення (або множина значень), тобто для кожного об'єкта $x \in U$ і кожного атрибута $a \in A$ існує відображення $f(x,a) : U \times A \to V_a$.

Набори даних, з яких необхідно здобути знання, називають навчальними вибірками. *Навчальна вибірка* – це будь-яка множина $X = \left(U, A \cup \{R\}\right)$, де R, $R \notin A$ – спеціальний атрибут, який називають класом об'єкта.

Задача індуктивного узагальнення полягає в тому, щоб за N-мірним простором ознак $A = \left\{A_1 \times A_2 \times ... \times A_N\right\}$ (A_i, $i = \overline{1,N}$, – множина значень і-ї ознаки об'єкта), множиною класів $R = \left\{r_1, ..., r_P\right\}$, $\forall a \in A \ \exists r_j$, $j = \overline{1,P}$, та за навчальною вибіркою X, $X \subseteq A \times R$, де $\forall x \in X$ визначений $r_j, j = \overline{1,P}$, побудувати *вирішальне правило* F, яке класифікує всі об'єкти з A так, що $\forall a \in A \ \exists F(a) = r, r \in R$, та правильно розпізнає клас об'єктів із навчальної вибірки: якщо $x = \langle a, r \rangle \in X$, тоді $\exists F(a) = r$.

Треба побудувати вирішальне правило F, яке:

- класифікує всі об'єкти з A: $\forall a \in A \ \exists F(a) = r, r \in R$;
- правильно розпізнає клас об'єктів із навчальної вибірки: якщо $x = \langle a, r \rangle \in X, a \in A, r \in R$, тоді $\exists F(a) = f(x, R) = r$.

Вирішне правило досить часто подають у вигляді дерева рішень, будь-який лист якого пов'язаний із певним рішенням – класом із R, а будь-який вузол, що не є листом, – з ім'ям атрибута A_i, значення якого треба запитати.

Існують незалежні напрямки розвитку систем, що використовують методи індуктивного здобуття знань: ID3 [184, 185], ACLS [295], CART [291]. Алгоритм ID3 розроблений для здобуття знань з великих обсягів слабоструктурованих даних, при його роботі час обчислень залежить лінійно від кількості введених прикладів,

атрибутів опису прикладів і вузлів у дереві рішень. Це вирізняє його від таких відомих алгоритмів побудови дерев рішень, як INDUCE [137, 160], SPROUTER [101], ROTH-P [225]. На вхід алгоритму надходить навчальна вибірка – набір класифікованих прикладів однакової розмірності.

Якщо навчальна вибірка містить приклади, у яких усі значення атрибутів однакові, а рішення різні, то введена інформація є недостатньою для побудови класифікаційного правила.

Якщо множина прикладів є порожньою, тоді можна довільно пов'язати її з будь-яким рішенням. Якщо всі приклади належать до одного класу, тоді будується єдиний лист дерева рішень, пов'язаний з цим класом. Інакше необхідно вибрати один з атрибутів і поділити множину атрибутів на підмножини залежно від значення атрибута і застосувати алгоритм до кожної з побудованих підмножин.

Алгоритм ID3 належить до незростаючих алгоритмів, тобто при додаванні до набору класифікованих прикладів певної кількості нових потрібно обробляти знову як старі, так і нові приклади. Якщо алгоритм є зростаючим, поточне визначення поняття переглядається, якщо необхідно, для кожного нового прикладу. До незростаючих алгоритмів відносяться також THOTH та UNDUCE. Як приклади зростаючих алгоритмів можна навести Candidate Elimination Algorithm [21], STAGGER, COBWER [295], Pocker Algorithm [225].

Істотним недоліком ID3 є те, що він будує класифікаційне правило тільки для двох класів. Алгоритм IID3M узагальнює ID3 на випадок довільної кількості класів і враховує ступінь доступності інформації про значення різноманітних атрибутів [339].

На кожному кроці роботи алгоритму опитується значення такого атрибута A_i, для якого

$$C(A_m) = \sum_i \sum_j \frac{C(A_m = a_{mi}, R = R_j)}{T(A_m)} = \\ = \max_s C(A_s) = \max_s \sum_i \sum_j \frac{C(A_s = a_{si}, R = R_j)}{T(A_s)}, \quad (5),$$

де

- $C(X,Y)$ – кількість інформації
  $C(X,Y) = \sum_i \sum_j p(X = x, Y = y) * \log p(X = x, Y = y);$
- $p(X=x,Y=y)$ – ймовірність одночасного здійснення подій $X=x$ і $Y=y$;
- $T(Am)$ – вартість отримання значення A_m.

Час, витрачений на класифікацію об’єкта за класифікаційним правилом, що будує алгоритм IID3M, у середньому не перевищує час класифікації об'єкта за будь-яким іншим класифікаційним правилом, побудований за тією ж навчальною вибіркою.

Критерій вибору атрибута (5), звичайно, дає задовільний результат, але розгалуження дерева рішень на кожному кроці для всіх можливих значень опитуваного атрибута призводить до ряду проблем: створюються дуже спеціалізовані правила, а кількість прикладів у вузлах зменшується, що знижує точність обчислень. Поділ значень атрибутів на дві підмножини викликає обчислювальну складність при виборі цих підмножин.

Ця проблема може бути вирішена формуванням булевих комбінацій пар атрибутів. Потрібно здійснювати поділ значень атрибутів на дві підмножини і будувати гілки для цих підмножин. Однак тут виникає обчислювальна складність: атрибут з r значеннями породжує 2^r можливих бінарних поділів. Алгоритм не може їх всі використовувати. У спеціальному випадку, коли є тільки два класи, потрібна тільки лінійна підмножина поділу.

Припустимо, що атрибути в середньому можуть приймати одне з r значень. Тоді існує $p*(r+1)^m$ можливих правил для прогнозування p класів. Звичайно, навчальна вибірка містить сотні прикладів з десятками атрибутів, тому програма не може обробляти увесь цей простір. Загалом, визначення мінімального набору правил, що покриває навчальну вибірку, має складність NP.

У зв'язку з цим пропонується алгоритм MID3 [343] – псевдобінарне узагальнення алгоритму IID3M, що дозволяє усунути розглянуті вище недоліки. Замість розгалуження для кожного значення, обраного за допомогою (5) атрибута, процедура дозволяє розгалужувати індивідуальні значення атрибутів і інші значення "купою" у вигляді однієї гілки, тобто в кожному вузлі дерева рішень атрибут умовно вважається бінарним і приймає два значення: "X" і "не X". Для того самого атрибута ці X можуть різнитися в різних вузлах. Вибір значення атрибута, що виділяється в окрему гілку, здійснюється з використанням інформаційної міри ентропії (6). Обирається те значення атрибута, що має найбільшу кількість інформації про результат:

$$C(a_{m_p}) = \max \sum_j C(A_m = a_{m_p}, R = R_j) \qquad (6)$$

Зростаючий алгоритм ID4 [225], розроблений Слімером і Фішером, будує таке ж дерево рішень, що і ID3, тільки у тому

випадку, коли у кожному вузлі дерева рішень існує атрибут, значно кращий за інші. Якщо ж відносний порядок критеріальних атрибутів у вузлі змінюється при введенні нового прикладу, тоді відкидаються всі піддерева, розташовані нижче цього вузла. Алгоритм отримує новий навчальний приклад і потім обновляє дерево рішень, що зберігається як глобальна структура даних. Недоліком алгоритму є те, що багато прикладів, які можуть бути вивчені за допомогою ID3, не можуть бути вивчені за допомогою ID4. Тому викликає інтерес алгоритм ID5R [225], розроблений Утгоффом, який гарантовано будує таке ж дерево рішень, що й ID3, для заданого набору прикладів. Як і ID4, цей алгоритм є зростаючим. Він зберігає достатню інформацію про дерево рішень, роблячи можливим зміну критеріального атрибута у вузлі. Однак замість відкидання піддерев нижче старого критеріального атрибута ID5 перебудовує дерево рішень так, що бажаний критеріальний атрибут потрапляє в корінь.

Незростаючі алгоритми [12, 185] використовують для задач, у яких обмежена множина навчальних прикладів, а зростаючі – для задач, у яких навчальні приклади поступають динамічно.

Нехай є скінченна множина атрибутів $\{A_1,...,A_n\}$, що впливають на результат і скінченна множина класів (рішень) $\{R_1,...,R_p\}$. Атрибут A_j приймає одне з t_j значень, $1 \le j \le m_n$, $\{a_{j_1},...,a_{j_{t_j}}\}$. *Навчальний приклад* – це набір з n+1 значення: $\langle b_1,...,b_n,c_f \rangle$, $c_f \in \{R_1,...,R_p\}$. *Навчальна вибірка* – це скінченна множина навчальних прикладів.

Результатом роботи алгоритму є дерево, у якому кожен лист пов'язаний з певним рішенням, а кожний вузол, що не є листом, – з атрибутом, значення якого потрібно запитати. Для мінімізації кількості таких запитів у дереві рішень на кожнім кроці визначається атрибут, повідомлення про значення якого несе найбільшу кількість інформації щодо результату, оцінюваного через ймовірність прийняття кожного з рішень залежно від значень атрибутів і ймовірність того, що цей атрибут матиме саме це значення.

Точні значення цих величин невідомі, тому вони апроксимуються на основі розглянутої множини прикладів.

Кількість інформації в повідомленні залежить від імовірності його отримання: чим більш імовірним є повідомлення, тим менша кількість інформації в ньому. Кількість інформації С про подію х у

події z $C(x,z) = \log_n(P(x,z))$, де функція P(x,z) – імовірність події x за умови, що відбулася подія z, а n – довільна константа.

Ширше застосування знаходить узагальнення алгоритму ID3, що дозволяє працювати не з двома, а з довільним (але, зрозуміло, скінченною і відомою заздалегідь) кількістю припустимих значень результату [339]. Точні значення ймовірностей подій з навчальної вибірки невідомі, вони апроксимуються на основі розглянутої множини прикладів:

$$P(A_m = a_i, R = r_j) = \frac{N(A_m = a_i, R = r_j)}{N(R = r_j)},$$

де функція N(q) – кількість прикладів у підмножині X, що розглядається на цьому кроці, для яких виконується умова q.

Кількість інформації, що очікують одержати в результаті з'ясування значення атрибута A, є сумою кількості інформації для кожного з можливих значень з урахуванням ймовірності одержання кожного із значень $C(A_m) = \sum_i \sum_j C(A_m = a_i, R = r_j) * P(A_m = a_i, R = r_j)$.

Необхідно знайти атрибут, для якого очікувана кількість інформації буде максимальною.

Це правило застосовується до навчальної вибірки ітеративно, доки у кожній підмножині прикладів не залишаються тільки приклади одного класу.

Запропонована функція не є єдиною можливою для даного класу задач. Хоча побудоване з її допомогою дерево рішень оптимальне щодо середньої довжини ланцюжка від кореня до листа, кількості вершин і дуг, для прикладних задач, у яких одержання значення ряду атрибутів пов'язане зі значними витратами часу або засобів, пропонується модифікована функція оцінювання, яка передбачає можливість урахування "вартості" опитування кожного атрибута, тобто складності одержання його значення.

Більшість алгоритмів індуктивного узагальнення (приміром, ID3, ID4, ID5, IID3M та MID3) розраховані на те, що як навчальна вибірка, так і дані в процесі консультації є повними. Але часто потрібно класифікувати об'єкти, проведення повного дослідження яких неможливо або недоцільно.

Дані вважають неповними, коли їх значення в даний момент невідоме, хоча може бути довизначене пізніше. Таке невідоме значення можна позначити спеціальною константою, і будь-яке входження його значення може бути замінене на конкретне з

множини припустимих. Запропонований Коддом метод "Null Values" [38] є адекватним способом формалізації неповних даних. На підставі таких даних однозначно класифікувати об'єкт не завжди можливо, але можна виділити підмножину класів, до яких може належати об’єкт при різноманітних засобах довизначення неповних даних. Для цього пропонуємо метод побудови такої підмножини – метод жовто-зелених гілок (ЖЗВ-метод) [182].

Для роботи з невідомими значеннями потрібно використовувати спеціальну тризначну логіку [129] з епістимічними значеннями істинності (T – "так", F – "ні", W – "можливо") і відповідними таблицями істинності для всіх логічних операцій. Застосування такої логіки до неповних даних поділяє їх на два класи: True-дані, значення яких завжди доступні, і Maybe-дані, конкретні значення яких можуть бути недоступні і пов'язані з пізнішим довизначенням.

Пропонується наступна технологія індуктивного узагальнення подібних даних [340, 344]. На першому етапі атрибути поділяється на дві множини відповідно до апріорних знань про їхню неповноту:

- m атрибутів, значення яких у процесі консультації завжди доступні, $m \le n$;
- k атрибутів, значення яких під час консультації можуть бути відомі, а можуть і не бути, $k=n-m$.

Формується матриця X', яка будується з матриці X шляхом переупорядкування стовпців таким чином, щоб перші m стовпців матриці X' відповідали True-даним.

На другому етапі навчальна вибірка розділяється на набір матриць: A з m стовпців і B[h] із k стовпцями. Матриця A складається з таких рядків, що для будь-якого її рядка A існує рядок матриці X', підрядком якого є підрядок рядка матриці A, що містить перші m атрибутів, і не існує іншого рядка матриці A, перші m значень якої збігаються із значеннями цього рядка.

Кожна з матриць $B[i], 0 < i \le h$, складається з таких рядків, що для будь-якого рядка матриці B[i] існує рядок матриці X', підрядком якого вона є і підрядок якого з перших m значень є підрядком і-го рядка матриці A.

Основний етап – побудова дерев рішень за допомогою алгоритмів індуктивного виведення для кожної з отриманих матриць.

До списку можливих значень кожного атрибута матриць B[i] додається ще одне значення – "невідоме" (значення атрибута відсутнє, не може бути отримано, невідомо точно тощо – дані типу Maybe). Це значення в ході консультації буде інтерпретуватися особливим чином

і при побудові дерева рішень не враховується (ситуація зі значенням атрибута "невідомо" можлива лише в процесі консультації).

На початку консультації знаходимося в корені дерева рішень. Розглянемо традиційну ситуацію, коли в процесі консультації оперуємо повними даними. У цьому випадку для кожного атрибута відомо його значення. Для наочності уявимо собі, що з вершини, зв'язаної з яким-небудь фактором, виходять N гілок, одна з яких відповідає відомому значенню і фарбується в зелений колір (дані типу True), а всі інші – у червоний. Якщо існує ланцюжок, що складається з зелених гілок, з'єднуючий кореневу вершину з яким-небудь листом дерева рішень, то в процесі консультації буде отриманий результат, що відповідає цьому листу. Очевидно, що такий результат – єдиний. На відміну від традиційної ситуації при консультації з неповними даними допускається вибір значення атрибута "невідомо". Всі гілки, які виходять з вузла, що відповідає атрибуту, значення якого невідомо, фарбуються в жовтий колір (дані типу Maybe), тобто є припустимими. Основна ідея ЖЗВ-методу полягає в тому, що в результаті такого фарбування дерева рішень у ньому виділяється піддерево, яке складається з гілок, пофарбованих у жовтий і зелений кольори. Всі листи цього дерева – рішення, які можуть бути отримані при різних варіантах довизначення неповних вхідних даних.

Для кожного з таких рішень можна отримати оцінку його ймовірності, яка визначається як відношення кількості гілок у жовто-зеленому піддереві, що приводять до цього результату, до кількості листів цього піддерева. Сума таких оцінок завжди дорівнює одиниці, тому цю характеристику можна вважати нормованою. Якщо в ході консультації отримана ймовірнісна оцінка результату, що дорівнює 1, то неповні вхідні дані дозволяють знайти однозначне рішення й тому не потребують довизначення [182].

Методи виведення знань за аналогією

У деяких задачах (наприклад, когнітивна психологія, розпізнавання природномовних текстів), що важко піддаються формалізації, в умовах неповної інформації застосовуються аналогії. Для ІС, які працюють у динамічно змінюваному середовищі, (приміром, пошукові системи Інтернету), типові ситуації, коли неможливо вивести необхідні знання дедуктивно з наявних даних, а набутого досвіду не вистачає для здійснення індуктивного узагальнення. Але іноді система обов’язково має приймати рішення та планувати свої дії навіть за таких складних умов. Тоді звертаються

до виведення за аналогією для прийняття рішення в конкретній ситуації. Таке виведення не є достовірним, а має характер припущення.

У виведенні за аналогією (традуктивне виведення) посилки стосуються одного об’єкта, а висновок – іншого. Можливість мислення за аналогією дозволяє людині відтворювати вже відомі механізми обробки інформації для нових ситуацій. Виведення за аналогією виконується згідно із загальними правилами, не орієнтованими на конкретні ПрО. Таке виведення зазвичай застосовують для формування початкових знань про ПрО, коли проблемно-орієнтовані правила для використання дедуктивного виведення ще не сформовані, а інформація про об’єкт дослідження не є достатньою для успішного застосування індуктивних методів.

Умовивід за аналогією – міркування, у яких з подібності двох об'єктів за деякими ознаками формують висновок про їх подібність за іншими ознаками. Приклад аналогії: посилка – "*З усіх плоских фігур рівної площі найменший периметр має коло*", висновок – "*З усіх тіл рівного об’єму найменшу поверхню має куля*". Інший приклад: "*Студент не зрозумів нічого з першого та другого розділів підручника*", висновок за аналогією – "*Мабуть, він і у третьому розділі нічого не зрозуміє*" (на відміну від висновку за індукцією – "*Студент взагалі нічого не роз*уміє").

Аналогія лежить в основі моделювання – методу вивчення об'єктів і процесів за їх моделями. В усіх методах, що базуються на аналогії, використовується схема виведення, запропонована Д.Пойя [305]: якщо об'єкт a_1 має ознаки b[1],b[2],...,b[k],...,b[n] і об'єкт a_1 схожий на a_2 за ознаками b[1],b[2],...,b[k], тоді a_2 також може мати ознаки b[k+1],...,b[n].

Умовивід за аналогією виконується за загальними правилами, тобто слугує інструментом формування первинних знань про ПрО, коли ще немає спеціалізованих предметно-орієнтованих правил для широкого застосування дедуктивного виведення. Крім того, ситуації, у яких можливі умовиводи за аналогією, зустрічаються частіше, ніж ситуації, коли можливе застосування методів індуктивного або дедуктивного виведення. Таким чином, методи використання аналогій у системах, що базуються на знаннях, універсальні і проблемно-незалежні. Однак їхні характерні недоліки пов'язані зі слабкою розробленістю питань вирішення протиріч, що виникають у базі знань, і з можливістю одержання недостовірних знань.

Прикладом системи, що реалізує традуктивне виведення, є програмний комплекс “Аналогія” [274]. Це експертна система-оболонка для розв'язання задач, пов'язаних із аналізом структури об'єкта досліджень. У цій системі реалізовано апарат виведення за аналогією на основі подання структурно-атрибутивних моделей знань у семантичній мережі.

Логіко-комбінаторні методи автоматичного навчання

У деяких ПрО внаслідок малого обсягу даних і їх неточного характеру виявилися досить ефективними методи логіко-комбінаторного типу (методи подібності, розходжень, залишків, індуктивного узагальнення Плоткіна). Індуктивні умовиводи – одні з найважливіших засобів формування правдоподібних тверджень, які є базисом цих методів. Метод подібності розглядає набір сутностей, кожна з яких має набір властивостей. На підставі посилань про наявність ряду властивостей робиться висновок про наявність інших властивостей сутності. Кількість посилань для того, щоб одержати потрібне узагальнення, не визначена. Їх має бути достатньо для того, щоб переконатися, що саме задана властивість є характеристикою класу.

$$\begin{array}{ll} & a, b[1] \Rightarrow "+" \\ \text{Якщо} & a, b[2] \Rightarrow "+" \quad \text{тоді} \quad a \Rightarrow "+". \\ & \ldots \\ & a, b[n] \Rightarrow "+" \end{array}$$

Метод розходжень є більш обґрунтованим. Його посилання враховують як наявність, так і відсутність властивостей сутностей. Він використовує посилання двох типів, яким можна поставити у відповідність множину позитивних прикладів (властивостей сутностей, що належать класу) і негативних прикладів (властивостей сутностей, що не належать цьому класу).

$$\begin{array}{ll} & a, b[1] \Rightarrow "+" \\ & \ldots \\ & a, b[n] \Rightarrow "+" \\ \text{Якщо} & b[1] \neg \Rightarrow "+" \quad \text{тоді} \quad a \Rightarrow "+". \\ & \ldots \\ & b[n] \neg \Rightarrow "+" \end{array}$$

Хоча формальний запис дає підставу вважати метод подібності частковим випадком методу розходжень, ці методи відрізняються зміщенням акценту з посилань типу "виводиться" у методі подібності на посилання типу "не виводиться" у методі розходжень.

Якщо для деяких сутностей апріорно інформація приналежності відома для окремих класів, то використовується метод залишків. Тобто з того, що деякі властивості притаманні окремим класам, і відомо, що деякі з перерахованих властивостей належать іншим класам, тоді решта властивостей належать до класів, що залишилися.

Якщо $a[1],..., a[n] \Rightarrow r[1],..., r[n]$

$a[2] \Rightarrow r[2]$

... тоді $a[1] \Rightarrow r[1]$

$a[n] \Rightarrow r[n]$

Методи індуктивних міркувань були сформульовані Д.С. Міллем ще в минулому столітті [310]. Правила подібності та розходження використовуються в методі формування гіпотез ДСМ, названого на честь Джона Стюарта Мілля [358].

Якщо в ситуації А приймається рішення r, то може бути висунута позитивна гіпотеза, що з а виводиться r, де а – фрагмент опису ситуації А. Якщо цей фрагмент входить в опис будь-яких ситуацій, що породжують рішення r, то ймовірність гіпотези збільшується. Ймовірність гіпотези збільшується і в тих випадках, коли а не входить в опис ситуацій, у яких приймаються рішення, відмінні від r. Відповідно, ймовірність гіпотези зменшується в тих випадках, коли а входить в опис ситуацій, у яких приймаються рішення, відмінні від r.

Приміром, множина T(0) включає всі гіпотези, що мають оцінку обґрунтованості, яка дорівнює 0, тобто всі гіпотези неістинні. Множина T(1) включає всі гіпотези, що мають оцінку обґрунтованості, яка дорівнює 1, тобто всі абсолютно достовірні гіпотези. Правила подібності та розходження на підставі позитивних і негативних прикладів, поданих у матриці, генерують нові гіпотези, для яких підраховується оцінка обґрунтованості з урахуванням кількості кроків виведення, характеристик правил, що використовуються при виведенні, а також оцінок обґрунтованості гіпотез, використовуваних як посилання.

Оцінки обґрунтованості гіпотез, знайдених раніше, перераховуються, при цьому множини T(0) і T(1) зберігаються. Усім гіпотезам, що підтвердилися і належать підмножині T(i/(n-1)), i=2,3,...,n-2, присвоюються оцінки (i+1)/n. Усім гіпотезам, що не підтвердилися і належать підмножині T(i/(n-1)), i=2,3,...,n-2, присвоюються оцінки (i-1)/n.

Крім правил подібності і розходження у ДСМ-методі застосовуються правила аналогії, за допомогою яких збільшуються

оцінки обґрунтованості тих гіпотез, що входять в особливу підмножину недовизначених гіпотез (підмножина T(1/(n-1))), і аналогічних за заданим критерієм новій виведеній гіпотезі.

Метод індуктивного узагальнення Плоткіна [178, 179] – процедура оберненої дедукції. На підставі множини фактів, що подаються мовою логіки предикатів першого порядку, даний метод створює індуктивну гіпотезу (у вигляді формули тієї ж мови), з якої факти можуть бути виведені дедуктивно. У цьому методі факти подаються у вигляді поєднання умов і результату. Виходячи з такого опису фактів, а також набору аксіом у методі Плоткіна генерується формула (пояснення), що є несуперечливим розширенням множини аксіом і забезпечує виведення усіх результатів із сукупностей умов. При побудові індуктивного узагальнення метод Плоткіна використовує алгоритм антиуніфікації [179]. Цей алгоритм задають найменше узагальнення пропозицій на основі заміни констант у цій пропозиції змінними. Найвужчим місцем цього підходу є питання про несуперечність розширення набору аксіом формулою узагальнення.

GUNA-метод автоматичного породження гіпотез надає математичний апарат для обробки експериментальних даних статистичного характеру. Метод використовує специфічні засоби формалізації: узагальнені квантори, емпіричні реляційні системи і відповідні емпіричні числення.

1.7. Експертні системи

Найвідоміші практичні застосування ШІ – експертні системи (ЕС), що являють собою інтелектуальні програми, здатні робити логічні висновки на підставі знань у конкретній ПрО та забезпечувати рішення специфічних задач на професійному рівні. Основу ЕС складає база знань ПрО.

Дослідники у галузі ЕС часто використовують також термін "інженерія знань", введений Фейгенбаумом як "залучення принципів та інструментарію досліджень із ШІ для рішення прикладних проблем, що вимагають знань експертів".

Експертна система – це складна ІС, що оперує знаннями певної ПрО з метою надання рекомендацій або вирішення проблеми. ЕС виконують функції експертів, компенсуючи їх недостатню кількість та оперативність при рішенні задачі.

Користувач – спеціаліст у певній ПрО з відносно низькою кваліфікацією, для якого призначена ЕС. *Інженер знань (ІЗ)* –

спеціаліст у ШІ, котрий виступає посередником між експертом та БЗ, яку він створює.

ЕС складається з таких компонентів (рис.1.7):

- *БЗ,* призначеної для зберігання експертних знань ПрО, які використовуються при рішенні задач ЕС;
- *БД,* призначеної для тимчасового зберігання фактів або гіпотез, які є проміжними рішеннями або результатом спілкування системи із зовнішнім середовищем;
- *машини логічного виведення* - механізму, що моделює хід міркувань експерта, оперуючи знаннями та даними з метою отримання нових даних із знань і інших даних, що містяться в робочій пам'яті;
- *інтерфейсу користувача,* призначеного для ведення діалогу з користувачами для отримання фактів, необхідних для процесу міркування;
- *підсистеми пояснень, що* дозволяє користувачеві розуміти процес отримання результату;
- *підсистеми здобуття знань,* призначеної для коригування і поповнення БЗ. У простому випадку це інтелектуальний редактор БЗ, який надає ІЗ можливість створення БД у діалоговому режимі. у складніших ЕС – засоби здобуття знань з БД, що містить неструктурований текст, графічну інформацію тощо.

Архітектура ЕС

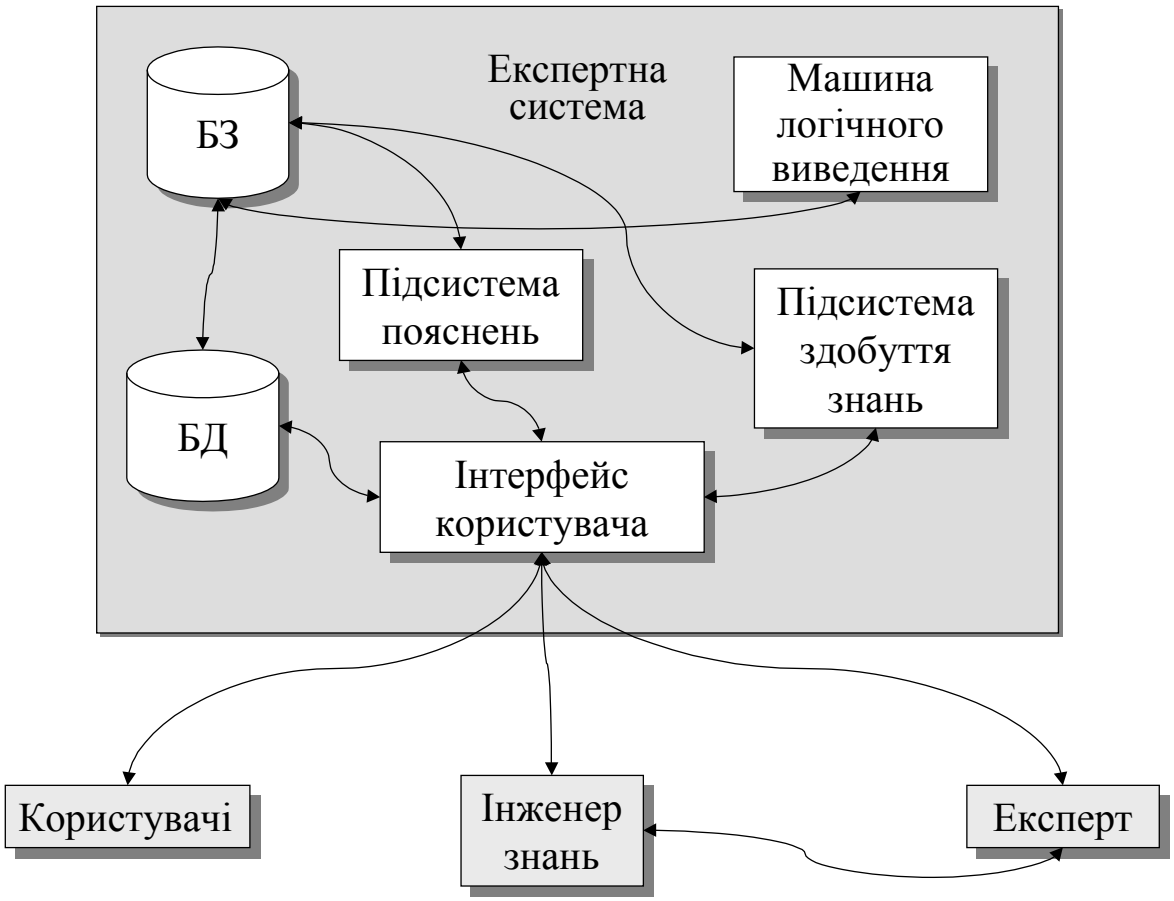


Рис.1.7. Архітектура експертної системи

База знань ЕС – це сукупність відомостей про ПрО, для якої розробляється ЕС. Для функціонування ЕС її БЗ має бути наповнена знаннями. Для цього запрошують *експертів* – висококваліфікованих спеціалістів у тій галузі, для якої розробляється ЕС. Їх завдання – формально описати всі свої знання, потрібні для функціонування ЕС.

У БЗ містяться знання двох типів:

- загальновідомі факти, явища, закономірності даної ПрО;
- набір емпіричних правил, відповідно до яких спеціалісти приймають рішення за умов невизначеності, неповноти та суперечливості інформації.

ЕС створюють на основі глибоких спеціальних знань про певну ПрО, отримувану від експертів**.** Система, що базується на знаннях (СБЗ), – це система, процес роботи якої пов’язаний з використанням символічного подання знань та правил їх обробки, а не чітких алгоритмів. ЕС відносяться до класу СБЗ, але, крім того, мають

давати конкретний результат за певний час та з потрібною достовірністю.

ЕС акумулює знання експертів для надання можливості використовувати їх менш кваліфікованим користувачам. Вона на основі обробки цих знань може давати інтелектуальні поради, приймати рішення на рівні експерта-професіонала, а також пояснювати процес знаходження того або іншого рішення.

ЕС властиві:

- спеціалізація у певній ПрО;
- використання БЗ;
- уміння пояснювати свої дії при розв'язуванні задачі та доводити їх обґрунтованість;
- здатність імітувати діяльність експерта;
- використання для розв'язування задач евристик – методів, що спираються на досвід та знання експерта.

ЕС мають широке практичне застосування, широко використовуються у науці (класифікація тварин і рослин за видами), медицині (постановка діагнозу, аналіз електрокардіограм, визначення методів лікування), техніці (пошук несправностей у пристроях, спостереження за польотом космічних кораблів і супутників), соціології, криміналістиці, лінгвістиці тощо. Проте складність та висока вартість ЕС, а головне – вузька спеціалізація – стримують їх впровадження.

У процесі розроблення такої БЗ можна виділити три основні фази: попередню, початкову і накопичувальну.

На *попередній* фазі ІЗ отримує від експерта або з інших джерел загальні відомості про ПрО (основні поняття, відношення, структуру даних) і формує загальне уявлення про принципи побудови ЕС, а потім обирає інструментарій для створення ЕС (приміром, порожню ЕС або мову подання знань) та середовища розроблення.

На *початковій* фазі ІЗ заповнює систему знаннями, що визначають організацію, структуру і спосіб подання БЗ.

Накопичувальна фаза характеризується набуттям основних знань про ПрО та передбачає виявлення неповноти, некоректності або суперечливості знань ЕС та здобуття знань, що усувають ці проблеми, а також надання цим знанням вигляду, зрозумілого ЕС.

Набуття знань передбачає спільну роботу ІЗ з експертами, які часто обґрунтовують свої висновки загальними концепціями, не виявляючи деталей, посилаються на інтуїцію та досвід, який

базується на великій кількості взаємозалежних фактів, закономірностей і навичок. Вирішити задачу створення БЗ ПрО можна за допомогою засобів автоматизованого здобуття знань.

Вимоги до ЕС:

- *компетентність:* у конкретній ПрО ЕС потрібно досягати того ж рівня, який мають спеціалісти – люди;
- ЕС має користуватися тими ж евристичними прийомами, так само глибоко і широко відображати *символьні міркування*;
- *глибина:* експертиза має вирішувати серйозні, нетривіальні задачі, що відрізняються складністю знань, використовуваних ЕС, або великим обсягом інформації;
- *самосвідомість:* ЕС має містити механізм пояснення того, яким чином вона приходить до розв'язання задачі.

Класифікація експертних систем

ЕС *інтерпретації даних* призначені для визначення семантики даних. Результати інтерпретації повинні бути погодженими і коректними. У таких системах нерідко використовується різноманітні методи аналізу даних. На сьогодні ці системи розвиваються в рамках напрямку, що одержав назву Data Mining — «здобуття» чи «заготівля» даних.

Діагностичні ЕС виконують функцію віднесення об'єктів до визначених класів. Галузь застосувань таких систем широка – від встановлення несправностей у технічних системах (технічна діагностика) до розпізнавання захворювань живих організмів, а також соціальних і природних аномалій.

ЕС *моніторингу* виконують задачу інтерпретації даних у реальному масштабі часу і сигналізують про вихід тих чи інших параметрів за припустимі межі.

ЕС *прогнозування* виводять ймовірні наслідки із заданих ситуацій. У прогнозуючих системах часто використовуються параметричні моделі, у яких значення параметрів «підганяються» під аналізовану ситуацію. Крім того, останнім часом для рішення задачі нерідко застосовуються інші підходи, зокрема, нейрокомп'ютерний підхід і різні алгоритми пошуку логічних закономірностей у структурах багатомірних даних.

ЕС для *планування* належать до об'єктів, здатних виконувати певні функції планування. У таких системах використовуються моделі поведінки реальних об'єктів для того, щоб послідовно вивести результати запланованої діяльності.

Щодо функціонування у реальному режимі часу ЕС поділяють на статичні, квазідинамічні та динамічні.

Статичні ЕС застосовуються для рішення задач, у яких БЗ і дані, що інтерпретуються, не змінюються з часом (приміром, ЕС діагностики несправностей автомобіля певної марки).

Квазідинамічні ЕС працюють у ситуаціях, що не міняються на деякому фіксованому інтервалі часу. Сюди належать, зокрема, ЕС в мікробіології, де лабораторні виміри технологічного процесу здійснюються кожні 4-5 годин і аналізується динаміка отриманих показників стосовно попередніх вимірів.

Динамічні ЕС обробляють дані, що постійно змінюються, часто в поєднанні з датчиками об'єктів, іноді в режимі реального часу з безупинною інтерпретацією даних, що надходять. Приклади: гнучкі виробничі системи, моніторинг у реанімаційних палатах тощо.

Можна класифікувати ЕС за ступенем інтеграції з іншими програмами.

Автономні ЕС застосовуються для рішення «експертних» задач у режимі консультації, коли не потрібно залучати додаткові методи обробки даних (розрахунки, моделювання тощо).

Гібридні ЕС поєднують стандартні пакети прикладних програм (приміром, пакети для аналізу даних, лінійного програмування). Вони являють собою інтелектуальні надбудови і виконують функції моніторингу стосовно відомого ПЗ.

Різновидом ЕС є *навчальні системи*, здатні давати обґрунтовані, методично ефективні для навчання пояснення з адаптивним ступенем деталізації по розглянутих рішеннях. Ці системи застосовують насамперед для професійного навчання майбутніх фахівців. Часто ЕС класифікують за призначенням.

ЕС, що *виконують інтерпретацію*, як правило, використовують інформацію від датчиків для опису ситуації. Приміром, це може бути інтерпретація показників вимірювальних приладів на заводі для визначення стану процесу. Такі ЕС мають справу не з чіткими символьними поданням проблемної ситуації, а безпосередньо з реальними даними. Це призводить до ускладнень, яких немає в інших ЕС, тому що їм доводиться обробляти недостатню, неповну, ненадійну або помилкову інформацію. Вони використовують спеціальні методи реєстрації характеристик безупинних потоків даних, сигналів або зображень і методи їх символьного подання.

ЕС, що *здійснюють прогноз*, визначають імовірні наслідки заданих ситуацій. Прикладами проблем, вирішуваних такими ЕС, є

прогноз збитків урожаю від деяких видів шкідливих комах, оцінка попиту на нафту на світовому ринку, прогнозування місця виникнення наступного збройного конфлікту на підставі даних розвідки. Системи прогнозування іноді використовують імітаційне моделювання, тобто програми, що відображають причинно-наслідкові взаємозв'язки в реальному світі, щоб генерувати ситуації або сценарії, які можуть виникнути за тих або інших вхідних даних. Можливі ситуації разом із знаннями про процеси, що породжують ці ситуації, утворюють передумови для прогнозу. На сьогодні розроблено порівняно мало систем для прогнозування, можливо, тому, що дуже важко взаємодіяти з імітаційними моделями і створювати їх.

ЕС виконують *діагностику,* використовуючи описи ситуацій, характеристики поведінки або знання про конструкцію компонентів, щоб установити ймовірні причини неправильного функціонування системи. Прикладами служать визначення причин захворювання за симптомами, що спостерігаються в пацієнтів, локалізація несправностей в електронних схемах і визначення відмов у системі охолодження ядерних реакторів. Діагностичні системи часто є консультантами, що не тільки ставлять діагноз, але і допомагають у налагоджуванні. Вони можуть взаємодіяти з користувачем, щоб надати допомогу в пошуку несправностей, а потім запропонувати порядок дій з їх усунення. Медицина – цілком природна галузь для діагностики, і, дійсно, у ній було розроблено більше діагностичних систем, ніж у будь-якій іншій ПрО, проте зараз багато діагностичних систем розробляють для застосування в інженерній справі та комп'ютерних системах.

ЕС, що *виконують проектування*, розробляють конфігурацію об'єктів, враховуючи обмеження ПрО. Прикладами можуть служити генна інженерія та синтез складних органічних молекул.

ЕС, зайняті *плануванням,* проектують дії. Вони визначають повну послідовність дій перед тим, як починається їх виконання. Прикладами можуть служити створення плану використання послідовності хімічних реакцій для синтезу складних органічних сполук або створення плану повітряного бою з метою нейтралізації певного чинника боєздатності ворога.

ЕС, що *виконують спостереження*, порівнюють справжню поведінку системи з очікуваною. Приміром, спостереження за показаннями вимірювальних приладів у ядерних реакторах мають виявляти аварійні ситуації, а оцінка даних моніторингу хворих у блоках інтенсивної терапії – небезпеку для життя людини. ЕС

порівнюють результати спостереження з даними, що притаманні стандартним ситуаціям. Такі ЕС за самою своєю природою мають працювати в режимі реального часу і здійснювати залежну як від часу, так і від контексту інтерпретацію поведінки об'єкта спостереження.

ЕС, що *навчають*, аналізують та коригують поведінку того, кого навчають. Ці системи створюють модель знань того, хто навчається, і модель того, як він ці знання застосовує до рішення проблеми. ЕС діагностує помилки і вказує на них, здійснює аналіз і будує плани виправлень цих помилок.

ЕС, що *здійснюють керування*, адаптивно керують поведінкою системи в цілому. Приклади: керування виробництвом і розподілом комп'ютерних систем або контроль за станом хворих при інтенсивній терапії. Такі ЕС мають містити компоненти спостереження, щоб відслідковувати поведінку об'єкта.

"*Порожні*" ЕС – це інструментальні засоби для побудови інших ЕС. Вони не містять конкретних правил ПрО. Прикладом такої системи є інструментальний комплекс ІндЕкс [340], що призначається для автоматичної розробки консультуючих систем. Цей комплекс містить бібліотеку алгоритмів індуктивного здобуття знань, підсистему візуалізації дерева рішень, інтелектуальний інтерфейс користувача, засоби пояснення та механізм інтерпретації дерева рішень. Система здатна обробляти неповні та нечіткі дані. За допомогою ІндЕкс були створені прикладні ЕС для оцінки якості геологічних досліджень, прогнозування масового розмноження шкідливих комах, оптимізації медико-біологічних досліджень осіб, що потерпіли внаслідок аварії на Чорнобильській АЕС, економічного прогнозування тощо [340, 344, 346].

Інструментальні засоби створення ЕС:

- символьні мови програмування, орієнтовані на створення ЕС і систем ШІ;
- мови інженерії знань, тобто мови високого рівня, орієнтовані на побудову ЕС (приміром, OPS-5, Пролог);
- системи, що автоматизують розробку систем ШІ, орієнтованих на знання;
- оболонки ЕС.

За зв'язком з реальним часом ЕС поділяють на наступні:

- статичні;
- квазідинамічні;

- динамічні.

Статичні ЕС розробляються для ПрО, в яких БЗ та дані, інтерпретовані нею, стабільні та незмінні. Приклад статичної системи ЕС для діагностики несправностей автомобіля.

Квазідинамічні ЕС здатні інтерпретувати ситуацію, що змінюється за певний інтервал часу. Приклад квазідинамічної системи: ЕС для обробки лабораторних вимірів технологічного процесу.

Динамічні ЕС здатні обробляти інформацію від датчиків у режимі реального часу. Приміром, вони використовуються гнучкими виробничими комплексами.

ЕС не взаємодіють безпосередньо з навколишнім середовищем: вони одержують інформацію не через датчики, а через користувача та інші ЕС, проте їх поведінка має багато спільних рис з агентами.

Інструментарій створення ЕС

Зараз існує різноманітне ПЗ, призначене для розробки ЕС. Ці системи, реалізовані численними мовами програмування, використовують різні платформи та операційні системи, орієнтовані на різноманітні типи задач та різняться за можливостями, які вони надають розробникам ЕС [349]. Розглянемо найпоширеніші з них.

За своїм призначенням та функціональними можливостями інструментальні засоби проектування ЕС, поділяються на такі категорії [283]:

- *оболонки ЕС.* Системи такого типу створюється зазвичай на основі певної прикладної ЕС, яка досить добре зарекомендувала себе. При цьому із системи-прототипу вилучають компоненти, що є специфічними для конкретної ПрО. Прикладами таких систем є EMYCIN та M.4, створені на основі MYCIN;
- *мови програмування високого рівня.* Інструментальні засоби цієї категорії позбавляють розробника від необхідності поглиблюватися в деталі реалізації системи. Одним з найбільш відомих представників цього класу – мова OPS5;
- *середовище програмування, що підтримує кілька парадигм*. Засоби цієї категорії включають кілька програмних модулів, що дозволяє користувачу комбінувати в процесі розробки ЕС різні стилі програмування. На основі цієї архітектури розроблено такі комерційні продукти, як KEE, KnowledgeCraft і ART;
- *додаткові модулі.* Засоби цієї категорії – це автономні програмні модулі, призначені для виконання специфічних задач. Приклад –

модуль роботи із семантичною мережею системи VT, що дозволяє відслідковувати зв'язки між значеннями раніше встановлених і нових параметрів проектування в процесі розробки проекту.

Об’єктно-орієнтований стиль програмування придатний для рішення проблем, що потребують деталізованого подання об’єктів ПрО і динамічних відносин між ними. Класичним прикладом застосування цього підходу є задачі моделювання. У таких програмах компоненти складної системи подаються через структури, які містять дані і функції, що моделюють поведінку відповідних компонентів. Першою мовою, у якій реалізована така ідея, є SmallTalk.

Розробка мови Common LISP пов'язана з потребою у стандартизації різноманітних діалектів LISP. Для задач ШІ розроблені мови LOOPS і FLAVORS – об’єктно-орієнтовані розширення LISP. Хоча на сьогодні ці мови практично не використовуються, але реалізовані в них базові ідеї успадковані багатьма мовами подання знань, що з'явилися пізніше. Приміром, CLOS (Common LISP Object System) підтримує такі властивості FLAVORS і LOOPS, як множинне успадкування, об’єднання методів і структурування метакласів.

OPS-сімейство мов подання знань реалізує продукційні системи з прямими виведеннями. Сюди входять різні реалізації мови OPS. Можливості керування вирішенням конфліктів у цих мовах різні. Наприклад, у мові OPS-5 пропонується вибрати одну з двох убудованих стратегій: LEX або MEA, а в OPS-83 необхідно явно вказувати, яке правило вибрати у кожній конфліктній ситуації (рис.1.8).

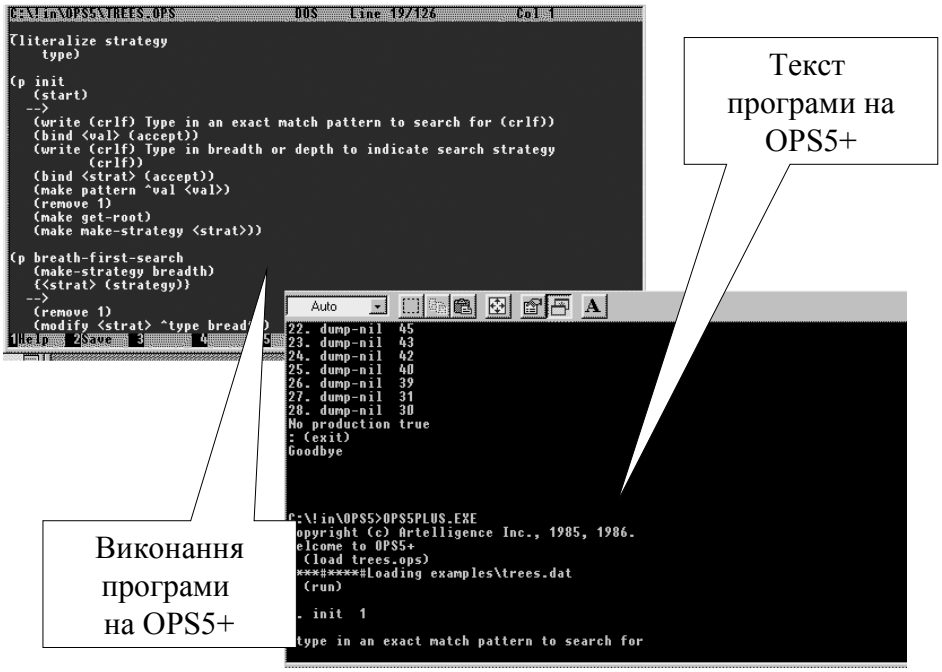


Рис.1.8. Виконання програми на OPS5+

OPS5+ [79, 183] – це високорівнева мова продукційного програмування, яка містить механізми подання знань і керування. Хоча ця система забезпечує основні потреби інженерії знань, вона не орієнтована на конкретні стратегії рішення задач або схеми подання знань. Система дозволяє програмісту використовувати символи і представляти відношення між ними, однак ці символи і відношення не мають заздалегідь визначених значень. Вони цілком залежать від правил, які описує програміст. Механізм керування інтерпретатора OPS5 – це простий цикл, деталі якого користувач розробляє сам відповідно до своїх потреб (рис.1.8).

Одна з цікавих реалізацій OPS-подібних мов подання знань – мова OPS-H. Її відмінні риси – поділ БЗ на сегменти, наявність можливостей керування виведенням, можливість підключення до системи будь-яких зовнішніх функцій, реалізованих мовою С. Для вирішення конфліктів використовують метазнання про порядок застосування правил, що описують ПрО.

Ця мова має наступні переваги порівняно із відомою мовою OPS5+:

- структурування програм на програмні модулі, кожний з яких може поділятися на програмні сегменти, що усуває проблему обмеження розміру БЗ, тому що передбачена можливість довантаження БЗ під час роботи програми;

- стратегія роботи машини виведення може програмуватися користувачем або обиратися з бібліотеки стратегій.

Текст програми на OPS-H представляється графічними структурами у вигляді P-графів [260], що дозволяє зменшити кількість помилок і підвищити продуктивність праці.

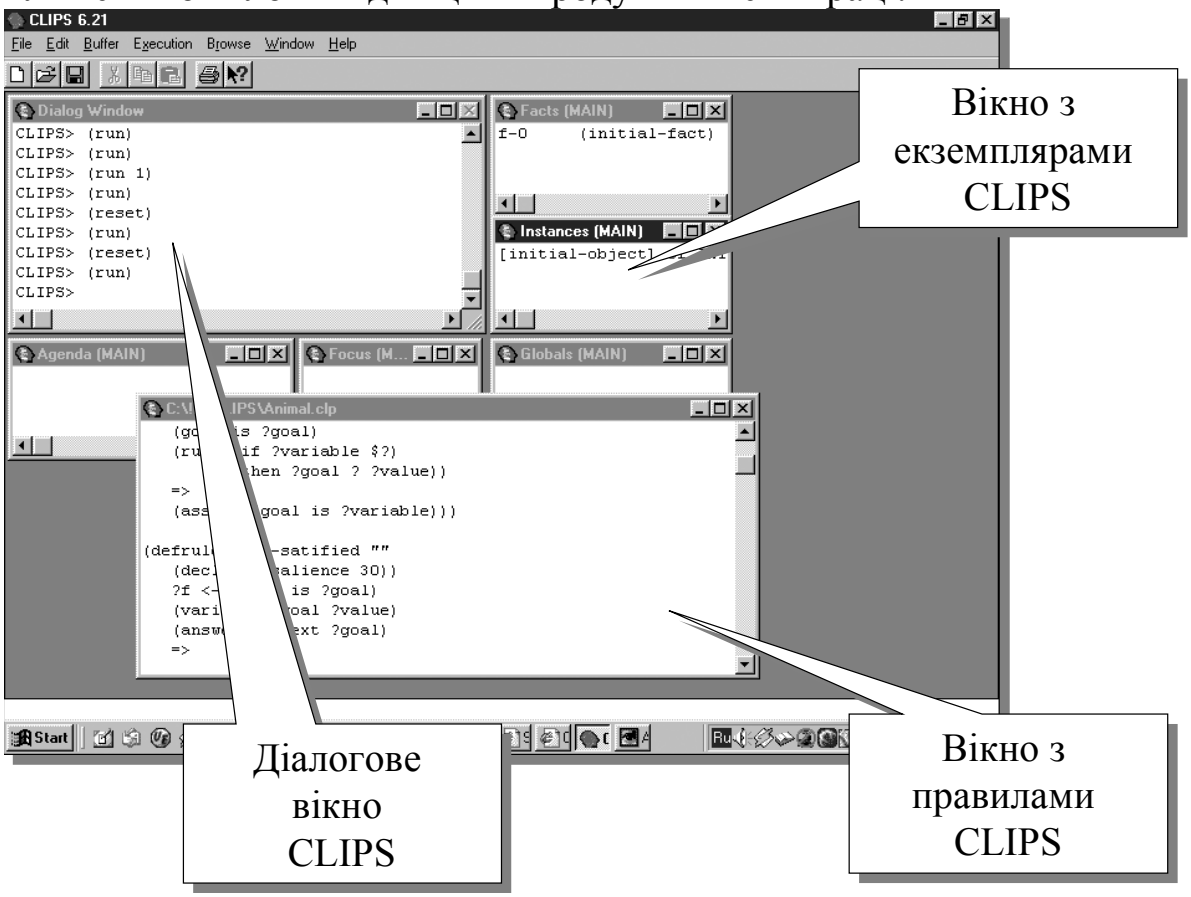


Рис.1.9. Інтерфейс користувача системи CLIPS

CLIPS (C Language Integrated Production System) [37] – OPS-подібна продукційна система, що використовує виведення від фактів до цілі. Механізм логічного виведення CLIPS включає супровід, динамічне додавання правил і стратегій вирішення протиріч. Основними елементами мови опису правил є бази фактів і правил. Машина логічного виведення співставляє ці факти та правила і встановлює, які саме правила потрібно активізувати. Вона легко вбудовується в інші прикладні програми і використовує об'єктно-орієнтовану мову COOL, що безпосередньо інтегрована з механізмом логічного виведення (рис.1.9).

DYNACLIPS (динамічні утиліти CLIPS) включає дошку оголошень, механізм динамічного обміну знаннями й інструментальні засоби для CLIPS v5.1 і v6.0. Це набір бібліотек, що може бути зв'язаний з CLIPS.

FuzzyCLIPS 6.02 [60] – оболонка експертної системи, що базується на правилах. Вона використовується для подання і керування нечіткими фактами і правилами. На додаток до функціональних можливостей CLIPS FuzzyCLIPS може обробляти нечіткі та неточні знання, складні міркування. Система використовує дві базисні концепції про неточності, нечіткість і невизначеність.

BABYLON – середовище для розробки ЕС, яке містить фрейми, моделі даних, Пролог-подібний логічний формалізм, а також мову для написання діагностичних прикладних програм.

MIKE – програмне середовище, розроблене для навчання, включає прямі і зворотні правила виведення від цілі до фактів з обумовленими користувачем стратегіями вирішення протиріч і фреймову мову подання знань зі спадковістю і "демонами", а також визначені користувачем стратегії спадкування. Порядок застосування правил у процесі виконання може відображатися графічно.

WindEx – повнофункціональна ЕС, що використовує виведення від фактів до цілі. Її модульна архітектура дозволяє користувачу заміняти модулі так, як це потрібно для розширення можливостей системи. WindEx містить процесор правил природною мовою, механізм логічного виведення та модулі БЗ.

KnowledgeWright [112] – це простий інструментарій для створення ЕС, який може підтримувати застосунки клієнтів, написані на C++, Java, Delphi, VB. KnowledgeWright є об'єктно-орієнтованою системою, яку можна використовувати безпосередньо за допомогою Web-інтерфейсів. Факти у KnowledgeWright подаються двома типами об’єктів. Об’єктами типу fact описуються тільки факти, відомі заздалегідь (при цьому їхній тип не вказується). Ті ж факти, значення яких обчислюються під час роботи системи, описуються об’єктами типу rule_set.

У перспективі ЕС будуть виконувати провідну роль на всіх фазах проектування, розробки, виробництва, продажу, підтримки і надання послуг. Технологія ЕС, що набула комерційного поширення, забезпечить революційний прорив в інтеграції застосунків з готових інтелектуально взаємодіючих модулів. Існує тенденція перетворення ЕС на мультиагентні системи.

Висновки

Розглянувши засоби подання знань і методи їх здобуття, розроблені протягом розвитку ШІ, можна зробити висновки про наявність ґрунтовної теоретичної бази для проектування

інтелектуальних автоматизованих систем, здатних вирішувати складні нетривіальні завдання у різних галузях діяльності людини. Такі системи здатні аналізувати інформацію на семантичному рівні і надавати користувачам орієнтовані на знання інформаційні послуги.

Аналіз галузей застосування ЕС дозволяє визначити перспективи використання ІАС та намітити тенденції їх розвитку для застосування в динамічному гетерогенному розподіленому середовищі. Це потребує адаптації та подальшого дослідження методів ШІ та створення нових інформаційних технологій. Найбільший інтерес з цієї точки зору на сьогодні представляють агентні технології, розглянуті у наступній главі.

Глава 2. Теоретичні основи інтелектуальних програмних агентів

Одна машина може виконати роботу сотні
звичайних людей, але жодна машина не
замінить одну видатну людину
Е.Хаббард

Програмні агенти (ПА) – нова парадигма програмування, яка дозволяє перейти на новий, більш інтелектуальний рівень взаємодії користувача з програмним і апаратним забезпеченням. Вона сприяє підвищенню ефективності праці та дозволяє користувачам доручити ІС виконання досить складних завдань. Необхідні інтелектуальні, інтерактивні та автономні програмні системи, які здатні до співпраці з користувачем для вирішення його задач. Такі складні задачі не можуть бути вирішені засобами одного наукового напрямку та потребують використання міждисциплінарного підходу, який містить технології з інформатики, штучного інтелекту, когнітивної науки та математики.

Розподілене керування даними та обчислювальними процесами в умовах глобальних та корпоративних мереж спонукало до створення нової концепції середовища функціонування ПЗ як середовища взаємодії мультиагентних систем (МАС), кооперації і конкуренції інтелектуальних агентів.

2.1. Основні властивості програмних агентів

Теоретичні передумови виникнення програмних агентів

Теорія ПА використовує результати, отримані у процесі досліджень інших наукових дисциплін: теорії керування, розподіленого ШІ та когнітивної психології. ПА – програмні сутності, здатні діяти автономно та цілеспрямовано у динамічному середовищі для того, щоб виконати завдання користувачів.

Формальний опис динамічних систем через апарат теорії керування дозволяє прогнозувати їх поведінку. Якщо ця теорія допомагає досліджувати взаємодію агента і середовища, в якому він функціонує, то питання про те, як у агента виникають цілі і наміри і які механізми далі приводять до їхньої реалізації у вигляді дій, можна вирішувати за допомогою методів когнітивної психології і, зокрема, теорії мотивації. Основні напрямки теорії мотивації:

- *процес формування наміру*: за множиною мотивацій, можливо, суперечливих, визначається результуюча мотиваційна тенденція, що стає базисом для формування наміру до дії;
- *процес формування бажань і дій:* виходячи зі сформованих на попередньому етапі намірів, визначаються та ініціюються дії.

Дослідження мислення та мотивації людської поведінки походять від Платона й Аристотеля, які визначили цей напрямок через три загальні категорії: пізнання, емоції і мотивації.

Левін в теорії особистості описує діяльність людини як функцію властивостей особистості та її уявлень про ситуацію, в якій здійснюється ця діяльність. Під впливом цих робіт Левіна створені такі теоретичні розробки, як ймовірнісно-значимі моделі, утилітарна теорія рішень, динамічна теорія дій тощо. Ідеї, пов'язані з дослідженням особистісних вирішальних факторів мотивації, були адаптовані дослідниками ШІ для моделювання потреб і цілей агентів.

Для досягнення своїх цілей агент має створювати план дій. Задача планування описується за допомогою початкового та цільового станів світу і множини операторів перетворення станів, а також обмежень на можливість виконання операторів. Потрібно визначити послідовність дій, які переводять систему з початкового стану до цільового. Планування – це пошук у просторі станів [73]. Дії агентів також описують множиною операторів разом з поданням очікуваних ефектів операцій, які вони виконують. У розробці інтелектуальних ПА використовують символьне подання знань, властивостей і цілей, а виконання плану розглядають як переходи між дискретними станами.

Зміну мотивацій агентів у часі моделює динамічна теорія дій. Її використання вимагає вирішення питань подання та розпізнавання мотивів і ситуацій, а також методів оброблення цієї моделі.

Ідея агентів запропонована у 50-х роках XX століття Мак-Картні та Селфриджем [135]. Ці фахівці називали агентом програмну систему, яка за наявності поставленої мети самостійно виконує певні операції і може при цьому запитувати й отримувати вказівки користувача, подані у термінах природної мови.

Мінський [141], Селфридж, Кей [110], Негропонт [154] використовують для позначення різних типів агентів такі терміни, як інтелектуальні агенти, інтелектуальний інтерфейс, персональні агенти, програмні агенти, мережні агенти тощо.

Нвана [161] поділяє дослідження агентів на два періоди: перший – з 1977 року і другий – з 1990 року. У першому періоді дослідження

ґрунтувалися на досягненнях у галузі розподіленого ШІ і були присвячені взаємодії між агентами, декомпозиції задач, координації і кооперації, вирішенню конфліктів. У другому періоді акцент досліджень змістився від моделювання міркувань (reasoning) до планування розподілених та віддалених дій (remote action).

Важливі ідеї, що визначили розвиток ПА у 1977 р., пов’язані з розробкою Хевітта моделі "*concurrent actor*", у якій програми розглядаються як співтовариство акторів – сутностей, які взаємодіють через повідомлення. Кожен актор складається з двох частин: 1) структури з адресами інших акторів, відомих цій сутності; 2) активної частини, що описує поведінку актора у випадку отримання повідомлення.

Актори є попередниками агентів. У них існують на базовому рівні виконавчі механізми, які використовують сучасні інтелектуальні ПА. Можна розглядати сучасні ПА як розвиток ідеї акторів на основі методів розподіленого штучного інтелекту та об’єктно-орієнтованого підходу.

Агент – це, насамперед, комп'ютерна програма. З цього випливають такі властивості, як коректність, повнота, ефективність, надійність. При цьому агент виконує певні функції людини, надаючи користувачу потрібні йому послуги.

Вимога неперервності й автономії викликана потребою в тому, щоб агент був здатний гнучко реагувати на зміни середовища без постійного втручання користувача. Крім того, агент, який працює в середовищі з іншими агентами і процесами, має бути здатним спілкуватися з ними.

Термін "агентно-орієнтоване програмування" (АОП) був запропонований Шохамом [208] для опису набору дій, необхідних для створення ПА – програмних сутностей, що функціонують безперервно й автономно в конкретному оточенні, у багатьох випадках – разом з іншими процесами й агентами. Можна розглядати агентний підхід як метафору проектування та моделювання розподілених систем.

Питання про те, яку комп'ютерну програму варто кваліфікувати як ПА, і досі не знайшло однозначного рішення. Різноманіття підходів і застосувань показує, що ПА стали одним з базових напрямків досліджень у галузі ІТ. Однак при цьому термін “програмний агент“ використовувався без якої-небудь спільної угоди про його значення. У результаті цього деякі програми стали називати агентами тільки тому, що вони, приміром, могли використовуватися

для подання завдань віддаленим комп'ютерам або були здатні переміщатися самостійно між ними. Тому потрібно дати більш точне визначення агентам.

Існують різні визначення ПА залежно від їх призначення та акцентування певних технічних властивостей. П.Маєс визначає автономні ПА як комп'ютерні системи, що існують у складному динамічному середовищі, сприймають зміни у ньому та діють автономно, реалізуючи набір цілей або задач, для виконання яких вони створені. Б.Хайєс-Рот відзначає, що інтелектуальні ПА виконують три функції:

- сприйняття динамічних умов середовища;
- дії у відповідь на такі умови;
- міркування для інтерпретації сприйняття, рішення проблеми та визначення реакції.

Таким чином, можна визначити ПА як автономну фізичну або віртуальну обчислювальну одиницю, що базується на:

- власних ресурсах – знаннях та вміннях;
- засобах сприйняття середовища (сенсорах) та впливу на це середовище (ефекторах);
- моделі середовища, заснованої на знаннях про нього.

ПА забезпечують наступні функціональні можливості:

- вирішення задач або досягнення певних цілей на основі наявних ресурсів та навичок;
- вибір рішення між альтернативами та виконання цього рішення у певному середовищі;
- спрямована взаємодія з іншими агентами та середовищем, у якому функціонує ПА.

Класифікація ПА за призначенням:

- інтерфейсні агенти;
- Інтранет-агенти;
- Інтернет-агенти;
- гетерогенні агенти.

Інтерфейсні агенти можна розглядати як персональні асистенти, які допомагають користувачу працювати з різними програмними засобами. Такі агенти спостерігають за діями користувача та намагаються запропонувати йому такі дії, які мають спростити його роботу. Вони здатні адаптуватися до індивідуальних особливостей та потреб конкретного користувача. Інтерфейсні ПА здатні до самонавчання, що приводить до ефективнішого виконання ними своїх функцій. Для самонавчання агенти використовують:

- спостереження за поведінкою користувача;
- отримання позитивних або негативних оцінок від користувача (зворотний зв'язок);
- безпосереднє отримання інструкцій від користувача;
- консультації з іншими агентами.

Представниками цієї групи агентів є такі системи, як A-Match та Verbal Software Robots.

Інтернет-агенти є найбільш численними представниками інтелектуальних ПА. Вони виникли як засіб обробки та транспортування інформаційних ресурсів Інтернету. На відміну від інтерфейсних агентів вони здатні не тільки створювати персоніфікований профіль користувача, але й відповідно класифікувати інформаційні ресурси.

Інформаційний ресурс (ІР) – це будь-яка інформація, що має цінність у певній ПрО та може використовуватися людиною (явно або через ПЗ) для досягнення певної мети. Переважна більшість ІР зараз може розглядатися як документи, проте існують й інші види ІР (детальніше – у главі 3) .

Інтернет-агентів можна поділити на дві основні групи: статичні та мобільні. Статичні ПА звичайно вбудовані у браузер. Приміром, такі агенти можуть сортувати електронну пошту, повідомляти користувача про події та повідомлення, які, за наявними в агента відомостями, можуть його зацікавити. Мобільні Інтернет-агенти менш поширені. Прикладом такого агента є Jasper. Цей агент здатний не тільки знаходити інформацію, цікаву для його користувача, але й повідомляти про неї інших агентів.

Робота Інтранет-агентів схожа на роботу Інтернет-агентів, проте має власну специфіку. Характерні задачі, завдяки яким Інтранет-агентів виділяють у окрему групу, такі:

- автоматизація бізнес-процесів підприємства;
- виконання послуг для користувачів, пов'язаних з використанням інформації з бази даних підприємства.

Гетерогенні агенти інтегрують функції двох або більше агентів, які належать до різних типів.

На відміну від класичних систем ШІ агенти не тільки пропонують рішення проблеми, але й реально діють. Сукупність причин, через які ПА виконує певні дії, називають мотивацією. Для досягнення своїх цілей ПА конструює план дій, які мають призвести до виконання поставленого перед ним завдання.

ПА можна характеризувати в термінах алгебри поведінки, що є неперервною алгеброю з наближенням і двома діями: недетермінованим вибором і приєднанням префікса [122]. Оточення – це агенти, які підтримують функцію вставки, що включає поведінку агента та оточення як аргументи функції і повертає як значення нову поведінку оточення.

Зараз визначилися два різних, проте пов'язаних один з одним підходи до визначення агента. Відповідно до першого підходу агент визначається головним чином своїми діями і не може бути цілком охарактеризований набором своїх атрибутів. У другому підході агент визначається атрибутами, якими він володіє.

Агент – це об’єкт, що сприймає середовище за допомогою сенсорів і діє в ньому за допомогою ефекторів [201].

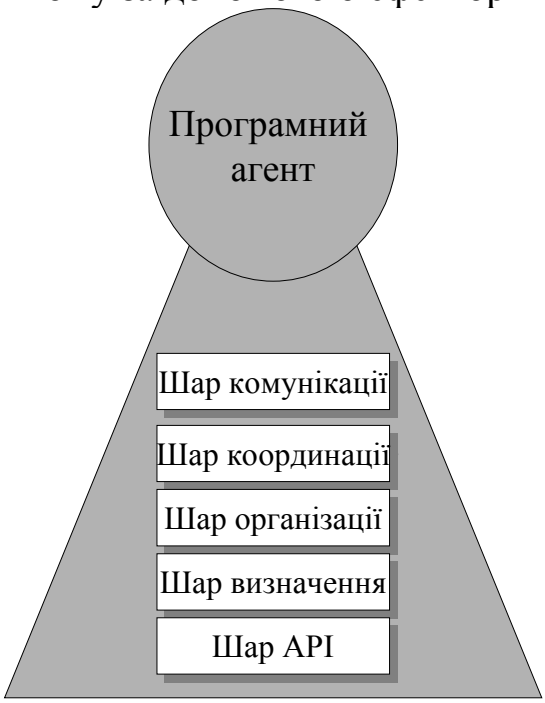


Рис.2.1. Багатошарова схема ПА

ПА функціонують у багатовимірному просторі. Вони складаються з кількох шарів (рис.2.1):

- комунікації;
- координації;
- організації;
- визначення;
- інтерфейсу API.

Шар комунікацій розглядає низькорівневі деталі взаємодії між ПА. На координаційному шарі подаються соціальні властивості ПА, технології координації та переговорів. На організаційному шарі ПА

визначається через відношення з іншими ПА, через ролі, які він виконує у взаємодії з цими агентами. На шарі визначення ПА визначається як автономна раціональна сутність, тобто в термінах механізмів міркування та навчання, цілей, ресурсів, здібностей, переконань тощо. Шар програмного інтерфейсу API пов’язує ПА з його фізичною реалізацією.

В обчислювальному середовищі кожний агент має свій життєвий цикл і власне ім'я. За допомогою використання доменів призначення агентів відбувається їх адміністрування.

Позначення імені агента забезпечує спосіб ідентифікації агента серед доменів інших груп агентів. Розробка програмних агентів вимагає застосування стандартизованих профілів агентів і методології розробки агентів для кожного з конкретних застосувань.

Розробка ПА вимагає застосування стандартизованих профілів агентів і методології розробки агентів для кожного з конкретних застосунків. Профіль платформи агента – це кортеж *<пароль, значення>*, що описує послуги і властивості платформи. Онтологія керування агента визначає словник і семантику опису можливості платформи агента. Атрибути ПА включають назву платформи агента, адресу, призначення сервісів агента, інформацію про підтримку в мережі, ступінь пріоритетів сервісів і їх застосування.

Онтологія керування агента визначає словник і семантику опису можливостей платформи агента. Атрибути агента включають назву платформи ПА, його адресу, призначення послуг, які може надавати ПА, ступінь пріоритетів цих послуг, інформацію про підтримку агента в мережі.

З погляду користувача, основна перевага використання агентів полягає в спрощенні взаємодії з програмою – користувачу досить поставити загальну задачу перед агентом, не вдаючись у подробиці того, як саме агент має її вирішувати. Якщо агент має можливості для рішення цієї задачі, він вирішує її сам, а інакше запитує необхідні йому послуги в інших агентів.

Агент – це програмний об’єкт, який:

- забезпечує виконання однієї або кількох корисних послуг;
- надає опис семантики цих послуг іншим ПА;
- здатний функціонувати автономно без безпосередніх вказівок користувача;
- може інтерактивно взаємодіяти з іншими ПА та користувачами.

В [153] наводиться таке визначення: агент – це компонент програмного або апаратного забезпечення ЕОМ, спроможний діяти, чітко виконуючи задачі від імені його користувача.

Функціональний зв'язок між агентом, середовищем, у якому цей агент функціонує, та його програмним кодом можна подати в такий спосіб: A=f(X), де

- A – агент;
- f – програма (функція);
- X – середовище.

Найпростіше визначення агента ґрунтується на моделі чорного ящика, що знаходиться у певному середовищі [197]. Агент описується як функція f, що обробляє інформацію від сенсорів і вхідні повідомлення (рис.2.2). Результат роботи агента – дії і вихідні повідомлення. Цей узагальнений підхід відповідає як біологічним, так і інтенсіональним моделям агентів. Розходження між цими моделями полягають у способі визначення f.

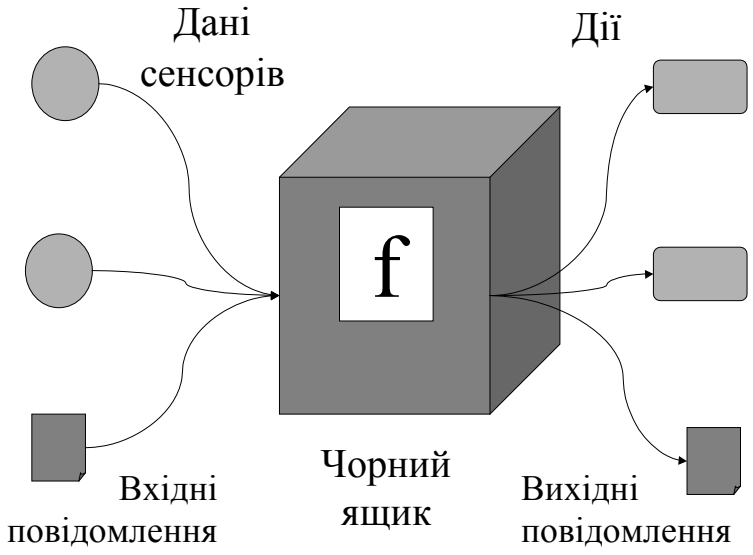


Рис.2.2. Визначення агента через модель чорного ящика

За визначенням FIPA (Federation of Intelligent Physical Agents) [76], *агент* – це об’єкт, що знаходиться в певному середовищі, від якого він отримує дані про події в цьому середовищі, інтерпретує їх і виконує команди, що впливають на середовище. Такий агент може містити як програмні, так і апаратні компоненти. FIPA – це міжнародна організація, створена в 1996 році з метою впровадження агентної парадигми для розробки практичних застосувань.

Нвана [153] називає ПА будь-яку програмну сутність з такими ознаками, як цілеспрямована поведінка, знання методів вирішення

проблеми для певної ПрО, автономна і проактивна діяльність від імені клієнта і здатність навчатися на досвіді спілкування з клієнтом або конкретною проблемою Це визначення висуває на перший план соціальні аспекти взаємодії між користувачем і агентом.

Можна визначати ПА через множину його атрибутів. ПА – термін, що дозволяє об'єднати множину більш специфічних і обмежених типів агентів, які мають деякі з таких атрибутів:

- *реактивність* (reactivity) – зміна своєї поведінки залежно від конкретної ситуації;
- *автономність* (autonomy) – самостійне виконання розпоряджень користувача без детальних інструкцій;
- *співробітництво* (collaborative behavior) – здатність працювати разом з іншими агентами для досягнення спільної мети;
- *спілкування на рівні знань* ("knowledge level" communication ability) – спроможність спілкуватися з людьми й іншими агентами мовою, близькою до природної;
- *здатність до логічного виведення* (inferential capability) – обробка абстрактного опису задачі з використанням апріорних знань про цілі і найбільш придатні методи їх досягнення, спроможність будувати моделі власної сутності, свого користувача, ситуацій та інших агентів;
- *безперервність у часі* (temporal continuity) – стійкість ідентифікації і положення протягом тривалого періоду часу;
- *персоналізація* (personality) – наявність персоналізованих значень атрибутів власної поведінки;
- *адаптивність* (adaptivity) – навчання й удосконалення на основі власного досвіду;
- *мобільність* (mobility) – здатність самостійно переходити з однієї платформи на іншу.
- *правдивість* – припущення про те, що агент не буде свідомо поширювати помилкову інформацію;
- *лояльність* – намагання робити те, що потрібно іншим агентам;
- *раціональність* – виконання тільки тих дій, які приводять до досягнення цілей.

Різні дослідники відповідно до тієї задачі, що постає перед ними, обирають різні множини атрибутів. Приміром, в [201] ПА визначається як комп'ютерна програма, що має наступні атрибути:

- *реактивність*;
- *автономність*;
- *прогресивність*.

Однією з основних характеристик агента є комунікабельність – здатність до гнучкого спілкування як з агентами, так і з іншими програмними компонентами. Унаслідок цього агенти відіграють важливу роль у досягненні інтероперабельності ПЗ, створеного незалежними розробниками в різний час [64]. Агентифікація дозволяє поширити цю властивість на довільне ПЗ. *Агентифікація* – перетворення довільного програмного забезпечення в ПА, приміром, у формі надбудови агентної оболонки над фрагментами програмного коду, яка забезпечує інтероперабельність цього коду [212].

Підкласом ПА є інтелектуальні агенти (ІА) (рис.2.3).

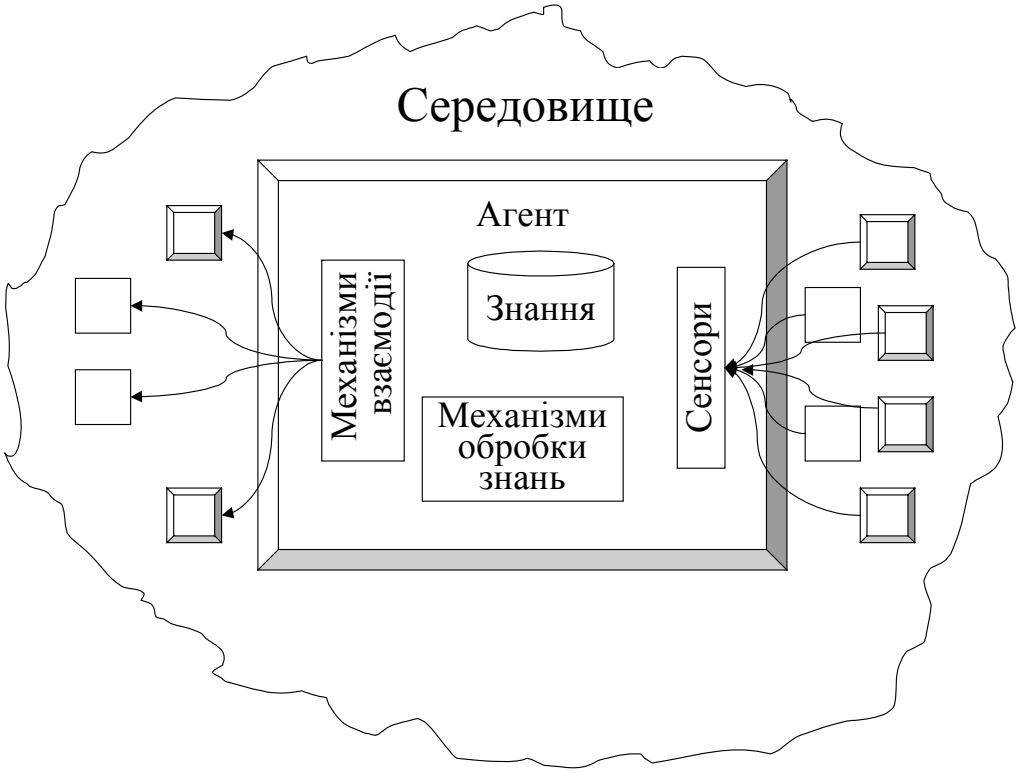


Рис.2.3. Загальна концептуальна схема інтелектуального ПА

Прийнято розрізняти вузьке ("сильне") та широке ("слабке") визначення терміну "інтелектуальний агент".

ІА у широкому розумінні – це ІС, що має такі ключові ознаки:

- *автономність* (autonomy) – функціонування значною мірою незалежно від втручання людини і контроль власних дій та внутрішнього стану;
- *соціальність* (social ability) – інтелектуальна та конструктивна взаємодія з іншими агентами та людьми шляхом обміну з ними повідомленнями деякою загальнозрозумілою мовою комунікацій;
- *реактивність* (reactivity) – сприйняття зміни середовища і вчасне реагування на них;

- *проактивність* (pro-activity) – здатність агента генерувати цілі і діяти раціонально для їх досягнення, а не тільки реагувати на зовнішні події.

Зазвичай, поведінка людини прогнозується і аналізується через такі атрибути відношень, як переконання, бажання, надії, побоювання тощо, які називаються інтенсіональними поняттями.

Деннет ввів термін інтенсіональних систем для опису сутностей, поведінка яких прогнозується шляхом приписування їм атрибутів переконання, бажання і раціональності [45], а Маккарті розглянув сферу застосування таких систем [135]. Чим менше відомо про систему і її структуру, тим більше корисні інтенсіональні пояснення її поведінки. Крім того, для досить складної системи (навіть за наявності повної інформації про неї) інтенсіональні пояснення її поведінки часто практично корисніші, ніж механістичні.

Більш строге ("сильне") розуміння терміну "інтелектуальний агент" вимагає наявності в агента *ментальних властивостей (інтенсіональних відношень),* до яких належать:

- *знання* (knowledge) – стала частина інформації агента про себе, середовище й інших агентів, що не змінюється в процесі його функціонування;
- *переконання* (beliefs) – знання агента, які можуть змінюватися в процесі його функціонування і ставати хибними, про поточний стан світу і про зміни в ньому, до яких має привести виконання дій агента;
- *бажання* (desires) – ставлення агента до майбутніх станів світу та переваги, які він надає одним з них порівняно з іншими (агент може мати несумісні та недосяжні бажання і тому не очікує, що усі вони мають бути досягнуті);
- *наміри* (intentions) – підмножина цілей, які може досягти обмежений у ресурсах агент, і засіб їх досягнення;
- *цілі* (goals) – несуперечлива підмножина бажань, досягнення яких агент прийняв як поточну стратегію поведінки;
- *зобов'язання* (commitments) стосовно інших агентів – завдання, що агент виконує за дорученням інших агентів у рамках кооперації та співробітництва.

Перші два поняття – переконання та знання – називають “точкою зору” (attitudes) агента, інші характеризують в англомовній літературі загальним терміном “pro-attitude”, сутність яких полягає у тому, що вони спрямовують дії та поведінку агента.

Інтенсіональні відношення поділяються на інформаційні (переконання та знання [113]) і перед-відношення (бажання та емоції [17], намір [40], зобов'язання, цілі тощо). Перші стосуються інформації, що має агент про світ, у якому він існує, тоді як перед-відношення – це відомості, які певним чином впливають на дії агента.

Перед- і інформаційні відношення тісно пов'язані, оскільки агенти можуть, наприклад, формувати наміри на основі наявної в них інформації про світ.

Деякі автори вважають, що інтелектуальному агенту мають бути притаманні такі властивості [241]:

- *мобільність* (mobility) – здатність переміщуватися телекомунікаційними мережами (локальними або глобальними) для досягнення своїх цілей;
- *доброзичливість* (benevolence) – готовність агентів допомагати іншим агентам та виконувати доручення користувача;
- *правдивість* (veracity) – властивість не повідомляти іншим агентам та користувачу інформацію, про помилковість якої йому відомо;
- *раціональність* (rationality) – здатність виконувати саме ті дії, що приводять до досягнення його цілей у рамках наявних у агента знань і переконань.

Використання агентної парадигми [135] потребує створення відповідної теоретичної бази. На сьогодні дослідження, спрямовані на розробку теоретичної моделі інтелектуальних ПА [243], ведуться у багатьох напрямках [8, 81, 92, 161].

Інтелектуальність ПА визначається його спроможністю міркувати і навчатися, наявністю моделі користувача, його потреб і механізму пошуку засобів їх задоволення [3, 96, 176].

У [265] вирізняють такі ознаки інтелектуальності агентів :

- автономне виконання своїх функцій;
- взаємодія з іншими агентами і користувачами;
- здатність стежити за оточенням;
- використання абстракції;
- застосування знання ПрО;
- адаптивність поведінки;
- навчання на власному досвіді;
- толерантність до помилок у вхідних сигналах;
- здатність працювати в реальному часі;
- спілкування природною мовою.

Використання агентів сприяє розвитку принципово нових інформаційних технологій.

В [256] наводяться характерні властивості ПА:

- здатність самостійно приймати рішення незалежно від користувачів;
- спроможність ініціювати дії інших агентів.

Модель ПА містить модель ПрО, модель користувача, засоби сприйняття (сенсори), засоби виконання дій (ефектори), цілі і планувальник дій на підставі цілей, моделі інших агентів і засоби взаємодії з ними

Модель ПрО, в якій функціонує ПА, відображає структуру та ієрархію об'єктів (наприклад, у вигляді онтології), на які спрямовані дії агентів та які впливають на способи досягнення його цілей. ПА має явно задану символьну модель світу, у якій рішення (наприклад, вибір дії) приймаються через логічні (або щонайменше псевдологічні) міркування, що базуються на відповідності між зразками та символьними маніпуляціями. Це приводить до двох проблем: як за час, упродовж якого інформація ще буде актуальною, адекватно описати реальний світ символами та як агентам обробити цю інформацію.

Модель користувача призначена для того, щоб ПА правильно інтерпретував завдання користувача та сповіщав про отримані результати у зручній та зрозумілій формі. У процесі роботи агент може поповнювати модель користувача, накопичуючи досвід взаємодії з конкретним користувачем (або класом користувачів) для підвищення ефективності своєї роботи.

Засоби сприйняття та засоби виконання дій ПА залежать від функцій, які агент має виконувати. Цілі агента визначають його дії відповідно до принципу раціональності: агент виконує тільки ті дії, які за наявною в нього інформацією та правилами її обробки приведуть до виконання хоча б однієї з його цілей. Планувальник дій агента визначає його дії та їх послідовність.

Моделі інших агентів потрібні ПА для того, щоб успішно взаємодіяти з цими агентами і обмінюватися з ними інформацією в процесі досягнення спільних цілей.

Таксономії програмних агентів

Для опису агентів більш ефективним способом, порівняно з аналізом усіх можливих комбінацій атрибутів, є різні схеми і таксономії. Приміром, Гілберт в [96] описує ПА в термінах тривимірного простору через дії, інтелект та мобільність (рис.2.4).

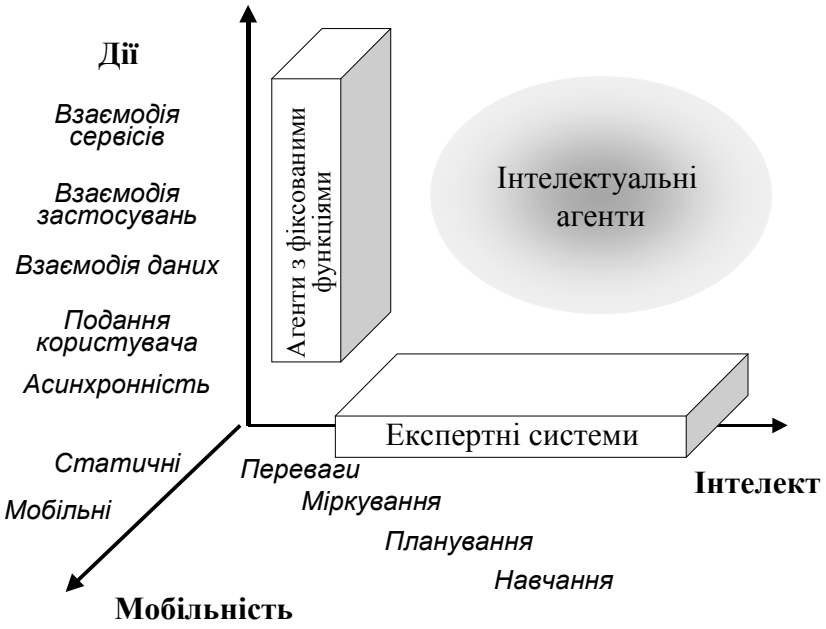


Рис.2.4. Класифікація агентів Гілбертом у термінах тривимірного простору дій, інтелекту та мобільності

Дамо визначення цих базових термінів за Гілбертом.

Дія – рівень автономності і повноважень агента, що може оцінюватися через природу взаємодії між агентом і іншими сутностями в системі. Агент має бути асинхронним. Його ступінь дії підвищується, якщо агент має певне уявлення про користувача. Більш розвинутий агент може взаємодіяти з даними, застосуваннями й іншими агентами.

Інтелект – рівень поведінки, пов'язаний з міркуваннями і навчанням. Здатність агента сприймати твердження користувача щодо цілей і виконувати його завдання пов'язана саме з цією якістю. Як мінімум, агент має вміти визначати пріоритет тверджень. Вищий рівень інтелекту потребує моделі користувача і здатність до міркувань. Ще більш інтелектуальними є системи, здатні навчатися за власним досвідом для того, щоб адаптуватися до свого середовища.

Мобільність – рівень самостійності агента при його переміщеннях мережею. Мобільні сценарії можуть формуватися на одній машині і надсилати запит для виконання на інший вузол мережі. Мобільні об'єкти переміщаються між вузлами під час виконання, переносячи накопичені ними дані.

Нвана [153] пропонує іншу типологію агентів, обравши такі параметри класифікації:

- мобільність,

- реактивність,
- первинні атрибути (автономність, співробітництво, навчання)

На основі цих характеристик Нвана розрізняє чотири типи ПА:

- співробітництва (collaborative);
- навчання і співробітництва (collaborative learning);
- інтерфейсу (interface);
- розумні (smart).

Додавши до цього опису такі характеристики, як роль, гібридні підходи, що поєднують кілька підходів в одному ПА, та вторинні атрибути (часова безперервність, вірогідність тощо), Нвана визначає ПА семи категорій:

- співробітництва;
- інтерфейсу;
- мобільні;
- інформаційні;
- реагуючі;
- гібридні;
- розумні.

Таксономія агентів, яку пропонують Франклін та Грассер [81], дозволяє описати більшість відомих прикладів ПА. Ця таксономія поділяє агентів за структурами керування, середовищем функціонування (приміром, база даних, файлова система, мережа Internet), за мовою, якою вони написані, і за застосуванням (рис.2.5).

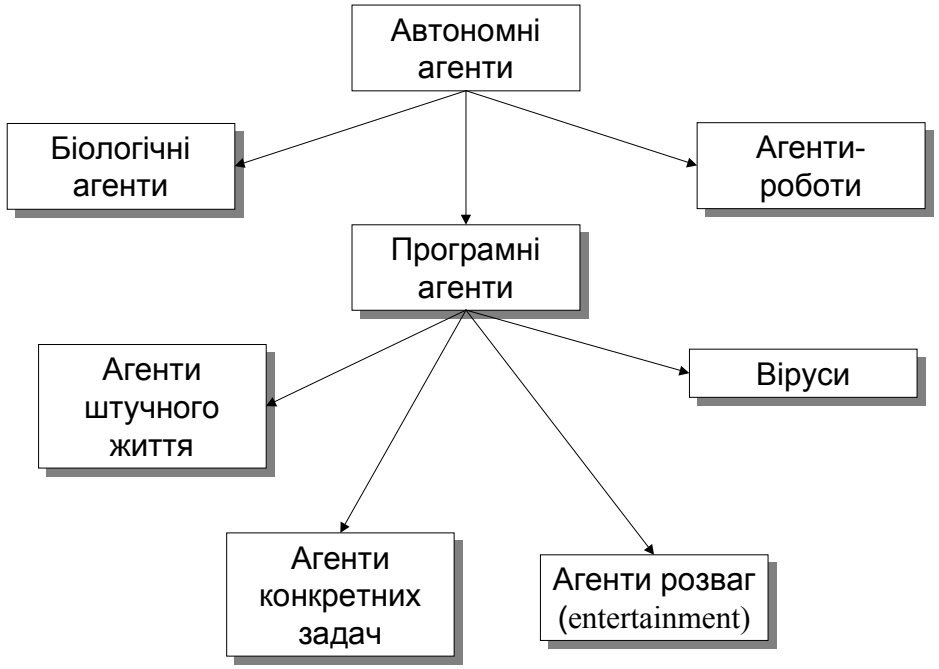


Рис. 2.5. Таксономія агентів Франкліна та Грассера

Подання інтелектуальних агентів через інтенсіональні відношення

Деякі дослідники вважають, що ПА доцільно описувати як інтенсіональні системи, використовуючи такі інтенсіональні терміни, як знання, наміри, обов'язки [208], емоції [17].

Як було зазначено вище, термін "*інтенсіональна система*" ввів філософ Деннет [45] для опису сутностей, поведінка яких може прогнозуватися через приписування атрибутів переконання, бажання і раціональної проникливості. Він виділяє різні типи інтенсіональних систем: інтенсіональна система *першого* порядку має припущення і бажання, але не має припущень і бажань щодо інших припущень і бажань; інтенсіональна система *другого* порядку більш складна – вона має припущення і бажання (а також інші інтенсіональні стани) як щодо своїх власних припущень і бажань, так і щодо припущень і бажань інших. Цю ієрархію інтенсіональностей можна продовжити.

Прийнято говорити про припустимість чого-небудь, про гіпотетичні події, цілі, яких можна досягти, здогади тощо. Значна частина тверджень може бути істинною або хибною залежно від обставин, поточного моменту, точки зору. У природних мовах модальності "*можливий*", "*необхідний*" і "*припустимий*" виражаються допоміжними дієсловами, такими як "*повинний*" і "*може*".

Можливість і необхідність називаються алетичними модальностями або модальностями можливості [304]. Так само, як квантори "*для всіх*" і "*існує*" вводилися в синтаксисі логіки першого порядку, можна побудувати формальну мову, використовуючи пари понять "*можливо*"/"*необхідно*" як квантори, що діють на формули. Логічна система, що базується на операторах "*можливо*" і "*необхідно*", називається *алетичною* логікою.

Для позначення модальності «необхідно» використовується символ □. Формула □F читається «необхідно, щоб F» або «F необхідно». Для позначення модальності «можливо» використовується символ ◊. Формула ◊F читається «можливо, що F». Один з цих операторів приймається за основний, а інший визначається через нього.

У природній мові вживаються й інші модальні форми, що важливо перенести в логіку. *Деонтична* логіка вводить модальності "дозволено", "обов'язково", що реалізують модальні мовні конструкції "дозволяється", "потрібно". *Епістемічна* логіка, або логіка знання, досліджує модальності "*знання*" і "*віри*", тоді як темпоральна логіка вводить модальності "*іноді*" і "*завжди*" (у "*майбутньому*" і

"*минулому*") разом з їх запереченнями "*часто*" і "*ніколи*". Через формальну подібність між цими логічними системами їх зазвичай вивчають разом. Для позначення сукупності цих логік використовують термін "*модальна логіка*". Найстарша серед них – *логіка можливого*. Саме її традиційно називають модальною.

Назва "модальна логіка" походить від того, що модальні логічні системи вводять такі оператори над логічними формулами, які дозволяють модифікувати їхню інтерпретацію. Приміром, у твердженнях "*Можливо, що F"*, "*А думає, що F"*, "*Часто правда, що F"*, "*У майбутньому, можливо, буде правда, що F"*, вирази, що передують логічній формулі F, є модальними операторами. Вони можуть відноситися до якої завгодно логічній формулі F. Значення істинності цих формул залежить не тільки від значення істинності формули F, яку вони містять, але і від моменту проголошення модальної формули (темпоральні логіки), від суб'єкта, що думає, що F, або вірить, що F (логіки віри), чи від необхідного, можливого або випадкового характеру певного факту (логіки можливого).

Існує подібність між визначеннями пар операторів («можливо»/«визначено», «іноді»/«завжди» тощо) і визначенням пари кванторів існування/загальності в логіці першого порядку. Модальний оператор спільності $\Delta\Diamond$ – це:

- ім'я модального оператора,
- вираз, що складається з імені модального оператора, за яким слідує список термів $t_i, i = \overline{1,n}$.

Модальний оператор існування ∇ визначається через заперечення модального оператора спільності: $\nabla F =_{def} \neg\Delta\neg F$.

Першою функцією модальної логіки була формалізація модальностей «можливість» і «необхідність». Інше її застосування – моделювання й аналіз парадигм «знання» і «віра». Для цього логічні системи використовують формальні мови з модальними операторами «віри» і «знання». Системи поєднують різні схеми аксіом і правила виведення для формалізації властивостей цих операторів. Модальні системи мають специфічну семантику. Різні модальні логіки є розширеннями логіки першого порядку. Зокрема, вони запозичають звідти аксіоми, правила виведення і теореми.

Нехай F – модальна мова висловлень, p і q – метазмінні, що позначають формули в мові F. Модальні оператори в F позначимо символами L і M (у модальній логіці необхідності і можливості це $\Box$ і $\Diamond$). Оператори загальності L і існування M зворотні: $Lp \equiv \neg M\neg p$.

У логіках віри і знання оператор L приймає відповідно значення "*передбачається"* і «**відомо**». Значення оператора M – "*протилежне не передбачається"* і "*протилежне не відомо"* відповідно.

Нормальна модальна система містить:

- множину всіх теорем логіки висловлень, область дії яких поширена на формули модальної мови висловлень F;
- схему аксіоми дистрибутивності $L(p \supset q) \supset (Lp \supset Lq)$, що позначається як K. Відповідно до тлумачення модальності "необхідність" схема K стверджує: "*якщо необхідно, що з p випливає q, тоді з необхідності p випливає необхідність q*";
- правило *modus ponens* $\frac{p, p \subset q}{q}$;
- модальне правило виведення необхідності $\frac{p}{Lp}$ ("p необхідно істинно" за умови, що "q істинно").

Можна отримати складніші модальні системи, розширюючи нормальну модальну систему такими схемами аксіом:

- *схема аксіоми знання* позначається як T. Схема T стверджує "*Те, що відомо, вірно*". Цю схему додають до нормальної модальної системи, щоб оператор L означав «відомо». Схема T відсутня в системі аксіом, що формалізують «припущення», тому що вони можуть бути помилковими. Нормальна модальна система, доповнена схемою T, позначається через KT;
- *схема аксіоми позитивної інтроспекції* $Lp \supset LLp$ позначається як 4. Якщо модальний оператор L означає «відомо», тоді схема 4 стверджує «Якщо мені відомо p, те я знаю, що відомо p». Якщо ж модальний оператор L означає «передбачається», то схема 4 означає «Якщо я припускаю, що p підтверджується, тоді я припускаю, що я припускаю, що p підтверджується». Описана схемою 4 можливість інтроспекції потрібна для формалізації досконалого інтроспективного інтелекту. Нормальну модальну систему, доповнену схемами аксіом T і 4, позначають KT4 або S4;
- *схема аксіоми негативної інтроспекції* $Mp \supset LMp$ позначається як 5 і в логіках знання і віри формалізує досконалу негативну інтроспекцію. Схема 5 еквівалентна такій схемі: $\neg Lp \supset L\neg Lp$. Коли оператор L означає «відомо», тоді схема 5 стверджує «Якщо я не знаю, що p підтверджується, тоді я знаю, що я не знаю, що p підтверджується». Якщо ж оператор L означає «передбачається», то вона твердить «Якщо я не припускаю, що p підтверджується,

тоді я припускаю, що я не припускаю, що p підтверджується». Ця властивість виражає досконале розуміння межі нашого знання чи віри. Нормальна модальна система, доповнена схемами аксіом T, 4 і 5, позначається KT45 або S5.

Вибір модальної системи залежить від поняття, що моделюється. Якщо потрібно охарактеризувати знання розумного суб'єкта, який володіє досконалою спроможністю до логічної інтроспекції відносно того, що "*відомо*" і що "*невідомо*", то слід вибрати модальну систему S5. Якщо потрібно моделювати припущення ідеально розумного суб'єкта (деякі припущення якого можуть виявитися помилковими, але який має досконалу спроможність до логічної інтроспекції щодо того, що він припускає і чого не припускає), то краще вибрати систему K45, яку називають також слабкою S5-системою.

Кожна з цих різних модальних систем ініціює властиве їй синтаксичне відношення виводимості. Воно позначається $|\!-_S$, де S – ім'я модальної системи.

Модальність збільшує виразність класичної логіки і дозволяє виявляти деякі поняття за допомогою специфічних операторів. Семантичний аналіз модального виразу залежить від параметрів, що неявно внесені цими операторами. Приміром, вирази мовою темпоральної логіки використовують модальні оператори з неявною змінною, що відображає часову еволюцію. Формула □p еквівалентна формулі $\forall t : p(t)$. Семантичний аналіз цієї формули має здійснюватися з урахуванням неявно припущеного параметра t у модальному операторі □.

Кріпке [114] пропонує специфічний метод семантичного аналізу – семантику можливих світів: модальна формула оцінюється через деякий "*універсум*" різних "*можливих світів*". Аналіз істинності модальної формули залежить від розглянутого можливого світу. Відношення доступності пов’язує можливі світи між собою.

Універсум W – це множина можливих світів, що пов'язані відношенням доступності R. Нехай a і b – два світи з універсуму W, тоді факт доступності світу b після світу a позначається aRb. Пара (W,R) називається структурою. Властивості відношення R приводять до різних схем модальних аксіом, що є загальнозначущими. Оцінка V – відображення з $W \times F$ у {Істина, Хибність}, що для кожного світу w з універсуму W зіставляє кожній пропозиційній константі з F певне значення істинності. Трійка (W,R,V) називається моделлю.

Для семантичної оцінки формул з множини F бажано визначити конкретний світ з універсуму W разом з розглянутою оцінкою V. Рекурсивно визначають відношення семантичного виведення |= між моделями і формулами мови. Запис (W, R, T) $|=_m$f означає, що f істинно у світі w для моделі (W, R, V). Основні правила:

- (W, R, V)$|=_w$ Істина;
- (W, R, V)$|\neq_w$ Хибність;
- (W, R, V)$|=_w$ f, якщо V(w, f) = Істина та f – пропозиційна константа з F';
- (W, R, V)$|=_w$ f$\supset$g, якщо (W, R, V) |=f тільки тоді, коли (W,R, V) $|=_w$ g;
- (W, R, V)$|=_w$ Lf, якщо для будь-якого світу x універсуму W при wRx маємо (W, R, V)$\backslash=_x$f.

Зміст останнього правила: формула Lf підтверджується у світі w для деякої моделі (W, R, V), якщо формула f підтверджується в усіх світах універсуму W, доступних зі світу w. Воно враховує бажані значення модального оператора L. Дійсно, у логіці необхідного Lp відображає необхідність формули p: "*p необхідно в даному світі"* інтуїтивно означає підтвердження p в усіх світах, що доступні з даного. З іншого боку, у логіці знання формула Lp означає "*p відомо"* і, отже (інтуїтивно), p підтверджується в усіх можливих світах, які можна подати на основі деякої множини знань і припущень.

З подвійності M ≡ ¬L¬ M безпосередньо випливає правило (W, R,VT)$|=_w$Mf, якщо існує світ x з універсуму W, такий, що wRx та (W, R, V)$|=_x$f. Щоб проаналізувати істинність модальної формули в кожному з можливих світів, потрібно ввести кілька понять.

Формула f з F *загальнозначуща* в моделі (W, R, V) тоді і тільки тоді, коли f підтверджується в усіх світах цієї моделі, тобто якщо V(w,f) = Істина для усіх світів w з W або у символьному записі (W, R, T)|=f.

Формула f з F *загальнозначуща в структурі* (W,R) тоді і тільки тоді, коли f загальнозначуща в будь-якій моделі (W, R, V). Символьний запис: (W, R)|=f.

Формула f з F *загальнозначуща* тоді і тільки тоді, коли f загальнозначуща в будь-якій структурі (W,R). Символьний запис: |=.

Вкажемо на зв'язок між семантичним відношенням |= і синтаксичним відношенням |–. Існує відповідність між вибором схем основних аксіом формальної системи і властивостями відношеннями

доступності між можливими світами семантичної характеристики цієї системи. Можна показати, що конкретизації:

- схеми аксіоми дистрибутивності, тобто схеми K, загальнозначущі;
- схеми аксіоми знання (схеми T) загальнозначущі в будь-якій структурі з рефлексивним відношенням доступності R;
- схеми аксіоми позитивної інтроспекції (схеми 4) загальнозначущі в будь-якій структурі з транзитивним відношенням доступності R;
- схеми аксіоми негативної інтроспекції (схеми 5) загальнозначущі в будь-якій структурі з евклідовим (відношення R називається евклідовим, якщо з aRb і aRc випливає bRc) відношенням доступності R.

З факту «f – формула F» доказово випливають твердження:

- $|-_K f$ тоді і тільки тоді, коли f загальнозначуща в будь-якій структурі;
- $|-_{KT} f$ тоді і тільки тоді, коли f загальнозначуща в будь-якій структурі з рефлексивним R;
- $|-_{KT4} f$ тоді і тільки тоді, коли f загальнозначуща в будь-якій структурі з рефлексивним і транзитивним R;
- $|-_{KT5} f$ тоді і тільки тоді, коли f загальнозначуща в будь-якій структурі з рефлексивним і евклідовим R.

Структура можливих світів характеризує семантику модальних систем залежно від властивостей відношення доступності. Приміром, якщо вибрати систему KT45 для аксіоматизації властивостей модального оператора L, тоді її семантична характеристика складається з множини можливих світів, пов'язаних між собою відношенням доступності R, що є відношенням еквівалентності (тому що R має бути рефлексивним, транзитивним та евклідовим).

Приписування переконань, свободи волі, намірів, свідомості, можливостей або бажань системі, що не є особою, дозволяє відображати знання про систему і допомагає зрозуміти структуру системи, її минуле та майбутню поведінку. Інтенсіональні відношення – абстрактні інструменти, які забезпечують зручний засіб опису, пояснення та прогнозування поведінки складних систем.

Цей підхід можна застосовувати як для систем, структура яких відома, але надто складна (приміром, для операційних систем), так і для сутностей, структура яких не повністю досліджена [135].

Інтенсіональні відношення [208] дозволяють описати ПА як систему та прогнозувати його поведінку [45]. Теорія агентів розглядає ПА як інтенсіональну систему, для опису поведінки якої потрібний несуперечливий опис з використанням підмножини ментальних понять. Важливий розділ цієї теорії – це подання динамічних аспектів

функціонування як окремого агента, так і співтовариства агентів за допомогою темпоральних логік.

Невелл здійснює аналіз ПА на рівні знань [157], вважаючи, що агенти можуть досягти своїх цілей, використовуючи наявні в них знання. Поведінка агентів на рівні знань описується за принципом раціональності: якщо ПА має знання про те, що одна з його дій приведе до однієї з його цілей, то він виконує цю дію. На рівні знань ПА, які володіють однаковими знаннями і мають однакові цілі, не розрізняються, навіть якщо вони мають різну фізичну структуру.

Переконання, як і деякі інші інтенсіональні відношення, є референційно непрозорими, тобто істинність висловлення "*А вважає Х*" залежить не тільки від істинності значення висловлення Х, але і від суб'єкта А (приміром, висловлення "*Іванов вважає, що Нью-Йорк є столицею США*" може бути істинним, якщо Іванов не обізнаний у географії). Тому класична логіка не придатна для їхнього опису.

Крім того, існують твердження, оцінка істинності яких невідома, але суб'єкт упевнений у можливості або неможливості такої оцінки. Приміром, якщо в Іванова є собака і він переконаний, що порода цього песика – пудель (тоді, як це такса), то істинність переконання Іванова можна оцінювати. Якщо Іванов не має собаки, то неможливо оцінити твердження щодо породи неіснуючого собаки.

Формалізація інтенсіональних відношень дозволяє створити теоретичну символьну модель ПА. Теорія можливих світів, запропонована Хинтіккою [103] дозволяє визначити семантику мови, що містить інтенсіональні оператори переконання. У цій семантиці ментальні поняття інтерпретуються через множину можливих світів і відношення досяжності між ними. З кожним можливим світом асоціюється певна теорія (множина формул і атомарних предикатів – фактів, про які відомо, що вони істинні).

Загальнозначущим критерієм відмінності знання від переконання є фіксована точка зору, представлена базовою системою можливих світів W. При цьому переконання агентів можуть бути охарактеризовані як множини можливих світів. Повні знання про навколишнє середовище в загальному випадку недоступні.

Агент здатний серед усіх взагалі можливих ситуацій виділити та відкинути ті, які суперечать наявній в нього інформації. Кожна з ситуацій, яка не була відкинута, називається світом і є можливою, виходячи зі знань агента. Хинтікка використовує термін "*епістемічна альтернатива"* для опису можливих, відповідно до переконань агента, світів. Ті твердження, що є істинними в усіх епістемічних

альтернативах агента, можна назвати його переконаннями. Можливі світи можуть бути включені в семантичну структуру логіки. Епістемічні логіки звичайно формулюються як звичайні модальні з використанням семантик, запропонованих Кріпке [114]. Розглянемо твердження типу *"A (R)ρ",* де

- A – суб'єкт;
- ρ – розповідне речення або твердження;
- R – відношення між A та твердженням ρ.

Якщо як відношення R розглянути переконання, то твердження щодо істинності *"A переконаний у ρ"*, означає, що ρ визнається істинним для A незалежно від дійсної істинності ρ. Передбачається, що існує зовнішній суб'єкт K, який встановлює істинність твердження "A *переконаний у* ρ". Отже, існують дві точки зору: A – внутрішнього і B – зовнішнього суб'єктів, тобто в семантиці можливих світів [103] необхідно розглядати дві множини можливих світів. Зафіксуємо базисну множину W, що відображає реальний стан світу і відповідає знанням зовнішнього суб'єкта K. У W оцінюються як немодальні, так і модальні твердження. U – допоміжна множина можливих світів, що подає знання внутрішнього суб'єкта A. У світах $u \in U$ оцінюються тільки немодальні твердження, що виражають судження внутрішнього суб'єкта про об'єкти. Світ $u \in U$ описує світ $\varpi \in W$ з погляду A [268].

Як загальнозначущий критерій відмінності знання від переконання береться фіксована точка зору, подана базовою системою можливих світів W. У такому випадку пропозиція "*A переконаний у ρ*" буде істинною у світі $\varpi \in W$ тоді і тільки тоді, коли ρ істинно в світі $u \in U$, співвіднесеному зі світом $\varpi \in W$.

Твердження "*A знає ρ*" істинно в ω тоді і тільки тоді, коли ρ істинно у світі ω і в співвіднесеному зі світом $u \in U$ світу ω. Це означає, що суб'єкт встановлює істинність ρ, при цьому ρ дійсно істинно.

ρ – класичний можливий світ, тобто в ньому виконуються закони класичної логіки, і він цілком визначений: будь-яке правильно побудоване висловлення мови в ньому або істинно, або ні. Але суб'єкт A не завжди має у своєму розпорядженні повну і не перекручену інформацію про світ, тому його уявлення про світ складаються зі здогадів і припущень. Через це для опису світів з U двох значень – "*істинно*" і "*хибно*" – недостатньо (приміром, існують твердження,

оцінка істинності яких невідома, але суб'єкт упевнений у можливості або неможливості такої оцінки).

Існує залежність між значеннями істинності й особистими пізнавальними модальностями (таб.2.1).

Таблиця 2.1

Залежність між значеннями істинності й особистими пізнавальними модальностями

Модальність	ω-світ	u-світ
A знає ρ	ρ істинно	ρ істинно
A переконаний у ρ	ρ будь-яке	ρ істинно
A помиляється в ρ	ρ будь-яке	ρ істинно
A сумнівається в ρ	ρ будь-яке	ρ істинно або ρ хибно
A не знає ρ	ρ будь-яке	ρ будь-яке

Багато дослідників розвивають альтернативні формалізми подання переконань, щоб уникнути труднощів, пов’язаних з логічним всезнанням, до яких приводить теорія можливих світів.

Ідея Левескьє полягає в тому, що агент має досить малу множину явних переконань і дуже велику (нескінченну) множину неявних переконань, яка складається з логічних наслідків явних переконань [123]. Для формалізації цієї ідеї він застосовує логіку, яка містить оператори явних і неявних переконань. Семантика оператора явних переконань подається в термінах ослабленої семантики можливих світів, доповненої ідеями, запозиченими із ситуаційної семантики, а семантика оператора неявних переконань – в термінах стандартного підходу можливих світів. Проте поняття ситуації, що використовується в цій логіці, ще слабше, ніж формалізоване порівняння з поняттям світу в можливих світах [195], а за деяких обставин ця модель призводить до нереальних припущень про спроможності агента міркувати.

Дедуктивна модель переконань обмеженого в ресурсах агента, яку запропонував Конолідж [93] базується на методах символьного ШІ. ПА мають два ключові компоненти: БД символьно поданих переконань і механізм виведення. Припускається, що агент застосовує правила виведення завжди, як тільки це можливо, щоб згенерувати дедуктивне замикання своїх переконань за правилами дедукції. Ця досить вдала модель системи переконань агента, на жаль, не завжди може застосовуватися на практиці.

Всі розглянуті вище формалізми фокусувались тільки на інформаційних відношеннях. Проте теорія агентів має відображати також динамічні аспекти, тобто те, як атрибути агента співвідносяться між собою. Наприклад, необхідно показати, як співвідносяться в агентах інформаційні і перед-відношення, як когнітивний стан агента змінюється в часі, як оточення впливає на когнітивний стан агента, і як інформаційні та перед-відношення агента приводять до виконання дій.

Море [142] формалізує логіку, яка містить модальності для знань, і динамічний апарат моделювання дій для вивчення передумов: дій – що агенту треба знати для того, щоб бути спроможним виконати певні дії. Такий формалізм дозволяє агенту, що має неповну інформацію про спосіб досягнення цілі, виконувати дії з метою досягнення цієї цілі.

Формалізм, запропонований Кохеном та Левескьє [39], який спочатку був використаний для створення теорії намірів (у розумінні "*я маю намір*..."), можна ефективно використовувати для аналізу міркувань агентів . Кохен та Левескьє використовують дворівневий підхід до формалізації намірів: спочатку вони задають логіку раціонального агентства, яка сортує відношення між базовими модальними операторами, а потім вводять над цією структурою кілька похідних конструкцій, що встановлюють часткову теорію раціональних дій. Одна з цих конструкцій – це стійка ціль. Агент має *стійку ціль* тоді й тільки тоді, коли:

1) він має ціль остаточно зробити x істинним і припускає, що x не є в даний момент істинним.

2) для того, щоб агент припинив спроби досягти ціль, має виконатись одна з таких умов: агент вважає, що x задоволено або ніколи не буде задоволено.

Модель агента, в якій використовуються інтенсіональні відношення (не обов'язково саме переконання, бажання та наміри), звичайно називають BDI-моделлю (Belief-Desire-Intention). Наведений вище аналіз публікацій свідчить про поширеність використання для моделювання поведінки ПА різних комбінацій інтенсіональних відношень. Наприклад, Кохен і Левескьє [39] використовують як базові відношення переконання і цілі (інші відношення визначаються через них), а Рао і Джоржеф [188] – переконання, бажання і наміри, тоді як модель агента, запропонована Братманом [24], замість переконань, бажань і намірів використовує практичніші відношення –

цілі і плани. Вибір комбінації інтенсіональних відношень визначається специфікою інтересів розробників моделі.

Перші спроби побудови моделі ментальних понять базувалися на мовах числення предикатів першого порядку, однак вони не виявилися вдалими.

При виборі формалізмів для опису ментальних понять потрібно вирішувати два класи проблем: синтаксичну і семантичну, а будь-який формалізм подання ментальних понять (як і для подання будь-якої іншої інформації) має враховувати два окремих аспекти: мову формалізації і семантичну модель.

На сьогодні існує відносно невеликий вибір підходів до опису синтаксису і семантики [241]:

- використання *метамов* (багатосортна логіка першого порядку з термами, що позначають формули інших мов, при цьому ментальні поняття подаються через предикати метамови);
- використання розширень відомих *модальних логік,* що містять спеціальні модальні оператори.

Внаслідок динамічного характеру функціонування агентів ці логіки мають доповнюватися засобами опису темпоральних аспектів ментальних понять[95].

Семантика можливих світів досить складна для застосування в практичних задачах. Інший варіант семантичної інтерпретації ментальних понять – інтерпретація символічних структур за допомогою поставлених їм у відповідність функцій (алгоритмів) і структур даних. Обидва підходи отримали свій розвиток і застосування в області штучного інтелекту.

Мова, що містить модальні оператори

Нехай $\Im$ – мова класичної пропозиційної логіки. Розглянемо $\Im^e$ – епістемічну мову, що містить скінчену множину пропозиційних змінних Var={p,q,r,...}, зв'язки & та ¬, дужки і сімейство узагальнених модальних операторів $\{B_i, D_i, I_i: i, i = \overline{1,N}\}$. F – множина формул мови $\Im$ така, що:

- якщо $\alpha \in Var$, тоді $\alpha \in F$;
- якщо $\alpha \in F$, тоді $\neg\alpha \in F$;
- якщо $\alpha, \beta \in F$, тоді $(\alpha \,\&\, \beta) \in F$.

Множина формул мови $\Im^e$ – така множина F^e, що:

- якщо $\alpha \in F$, тоді $\alpha \in F^e$;
- якщо $\alpha \in F^e$, тоді $B_i\alpha \in F^e$, $i = \overline{1,N}$;
- якщо $\alpha \in F^e$, тоді $D_i\alpha \in F^e$, $i = \overline{1,N}$;

- якщо $\alpha \in F^e$, тоді $I_i\alpha \in F^e$, $i = \overline{1,N}$.

Інші зв'язування (V, $\supset$, $\equiv$) визначаються традиційно. Множину F^e будемо називати базовою, а множину F – допоміжною.

Для мови $\Im^e$ пропонується наступна семантика можливих світів. Упорядкована четвірка $\wp = \langle W, U, s, \varphi \rangle$ – модель мови $\Im^e$ тоді і тільки тоді, коли:

- W – непорожня базова множина можливих світів;
- U – непорожня допоміжна множина можливих світів;
- s – функція, що відображає множину W у множину U;
- φ – функція приписування, така, що для будь-якого $p \in Var$ $\varphi(p) = \langle \varphi_o(p), \varphi_T(p), \varphi_F(p) \rangle$, де $\varphi_o(p) \subseteq W$ і $\varphi_T(p) \subseteq U$ – множина світів, в яких ρ істинно, а $\varphi_F(p) \subseteq U$ – множина світів, де ρ хибно.

У системі світів U має місце наступний принцип несуперечності: $\varphi_T(p) \cap \varphi_F(p) = \varnothing$.

Допоміжна множина світів U описує умови істинності формул немодальної мови $\Im$. Для складних формул $\Im$ має місце наступне:

- $\varphi_T(\alpha \,\&\, \beta) = (\varphi_T(\alpha) \cap \varphi_T(\beta))$;
- $\varphi_T(\alpha \vee \beta) = (\varphi_T(\alpha) \cup \varphi_T(\beta))$;
- $\varphi_T(\alpha \supset \beta) = (\varphi_F(\alpha) \cup \varphi_T(\beta))$;
- $\varphi_T(\neg\alpha) = \varphi_F(\alpha)$;
- $\varphi_F(\alpha \,\&\, \beta) = (\varphi_F(\alpha) \cup \varphi_F(\beta))$;
- $\varphi_F(\alpha \vee \beta) = (\varphi_F(\alpha) \cap \varphi_F(\beta))$;
- $\varphi_F(\alpha \supset \beta) = (\varphi_T(\alpha) \cap \varphi_F(\beta))$;
- $\varphi_F(\neg\alpha) = \varphi_T(\alpha)$.

Визначимо поняття істинності і хибності в допоміжній системі U при даному приписуванні, для $\alpha \in F$, $u \in U$:

- $[\alpha]^{\varphi, u} = T \Leftrightarrow u \in \varphi_T(\alpha)$;
- $[\alpha]^{\varphi, u} = F \Leftrightarrow u \in \varphi_F(\alpha)$.

Нехай $\rho \in Var$, α і β – довільні формули епістемічної мови $\Im^e$, γ – довільна формула мови $\Im$. Визначимо поняття істини в даному світі з базової множини W при даному приписуванні:

- $[\rho]^{\varphi,w} = T$, якщо $w \in \varphi_o(\rho)$, у протилежному випадку $[\rho]^{\varphi,w} = F$;
- $[\alpha \,\&\, \beta]^{\varphi,w} = T$, якщо $[\alpha]^{\varphi,w} = [\beta]^{\varphi,w} = T$, у протилежному випадку $[\alpha \,\&\, \beta]^{\varphi,w} = F$;
- $[\neg\alpha]^{\varphi,w} = T$, якщо $[\alpha]^{\varphi,w} = F$, у протилежному випадку $[\neg\alpha]^{\varphi,w} = F$;
- $[B_i\gamma]^{\varphi,w} = T$, якщо $[\gamma]^{\varphi,s(w)} = T$, у протилежному випадку $[B_i\gamma]^{\varphi,w} = F$.

Базова система світів W забезпечує виконання класичних умов істинності для формул епістемічної мови $\Im^e$. Формула $\alpha \in \Im^e$ істинна в

моделі $\wp$ = <W, U, s, φ> тоді і тільки тоді, коли для кожного w ∈ W, кожне $\varphi[\alpha]^{\varphi,w} = T$.

Інтенсіональна модель ПА

Пропонуємо модель ПА, яка формалізує інтенсіональні відношення знання, переконання, помилки, намірів і цілей і зв'язки між ними для опису поведінки агентів [336]. Введемо відповідні модальні оператори.

Модальний оператор Bρ означає "*хтось переконаний у ρ*". У випадку N суб'єктів, які пізнають, можна ввести індекс $i, i = \overline{1, N}$, де $B_i\rho$ означає "*i-й суб'єкт переконаний у ρ*".

Нехай α і β належать до допоміжної множини формул F, γ і δ – формули базової множини F^e, $i = \overline{1, N}$. Наступні схеми аксіом і правило виведення описують модальний оператор переконання:

- A1. $B_i(\alpha \,\&\, \beta) \equiv (B_i\alpha \,\&\, B_i\beta)$;
- A2. $B_i(\alpha \vee \beta) \equiv (B_i\alpha \vee B_i\beta)$;
- A3. $B_i(\alpha \to \beta) \equiv (B_i\alpha \to B_i\beta)$;
- A4. $B_i \neg(\alpha \,\&\, \beta) \equiv (B_i \neg\alpha \vee B_i \neg\beta)$;
- A5. $B_i \neg(\alpha \vee \beta) \equiv (B_i \neg\alpha \,\&\, B_i \neg\beta)$;
- A6. $B_i \neg(\alpha \to \beta) \equiv (B_i\alpha \,\&\, B_i \neg\beta)$;
- A7. $B_i\,\alpha \to \neg B_i \neg\alpha$;
- A8. $B_i\,\alpha \to B_i \neg \neg\alpha$;
- A9. $B_i \neg\neg\alpha \to B_i\alpha$.

Правило виведення "*modus ponens*": якщо γ, γ -> δ, тоді δ.

Введемо в мову $\mathfrak{I}^e$ оператори, що характеризують пізнання: оператор знання К та оператор помилки Е. $K_i\alpha$ означає "*i-й суб'єкт знає α*", а $E_i\alpha$ – "*i-й суб'єкт помиляється, припускаючи α*".

- A10. $K_i\alpha \to (B_i\alpha \,\&\, \alpha)$.

У схемі А10 використаний знак імплікації, оскільки якщо при визначенні помилки досить указати на хибність того, що суб'єкт приймає як істинне, то з того факту, що істинно для суб'єкта й істинно у фіксованій системі, ще не можна зробити висновок про те, що суб'єкт знає про це: $(B_i\alpha \,\&\, \alpha)\&\neg\, K_i\alpha$.

- A11. $E_i\alpha \equiv (B_i\alpha \,\&\, \neg\alpha)$.

Має місце транзитивне відношення досяжності:

- A12. $K_i\alpha \to K_i\, K_i\alpha$.

Ця аксіома описує процес дослідження індивідуумом своїх переконань: він усвідомлює те, що він знає.

- A13. $B_i\alpha$ -> $К_i$ $B_i\alpha$.

Ця аксіома говорить про те, що агент усвідомлює, чого він не знає.

Введемо модальний оператор цілі. Твердження, що "А бажає ρ" істинно, означає, що ρ визнається кращим для А незалежно від дійсних можливостей А та досяжності для нього ρ.

Суб'єкт А може мати несумісні бажання, і він не повинний думати, що його бажання досяжні. Тому введемо більш вузьке поняття – ціль. Ціль – сумісна підмножина бажань. Dρ означає "хтось має ціль ρ". $D_i\rho$ означає "і-й суб'єкт має ціль ρ". Наступні схеми аксіом і правило виведення описують модальний оператор цілі:

- B1. $D_i(\alpha \,\&\, \beta) \equiv (D_i\alpha \,\&\, D_i\beta)$;
- B2. $D_i(\alpha \vee \beta) \equiv (D_i\alpha \vee D_i\beta)$.

Цілі (на відміну від бажань) не суперечать одна одній:

- B3. $D_i\neg\alpha \equiv \neg D_i\,\alpha$;
- B4. $D_i\,\neg(\alpha \,\&\, \beta) \equiv (D_i\,\neg\alpha \vee D_i\,\neg\beta)$;
- B5. $D_i\,\neg(\alpha \vee \beta) \equiv (D_i\,\neg\alpha \,\&\, D_i\,\neg\beta)$;
- B6. $D_i\,\alpha \equiv D_i\neg\,\neg\alpha$;
- B7. $D_i\,(\alpha \to \beta) \to (D_i\,\alpha \to D_i\,\beta)$;
- B8. $B_i(\alpha \to \beta)\&D_i(\alpha) \to D_i\,\beta$;
- B9. $B_i(D_i\,\alpha) \equiv D_i(\alpha)$.

Твердження, що "А має намір ρ" істинно, означає, що "А бажає ρ" і "А думає B(ρ)", де B(ρ) – спосіб досягти ρ. Iρ означає "хтось має намір ρ". $I_i\rho$ означає: "*і-й суб'єкт має намір ρ*". Наступні схеми аксіом і правило виведення описують модальний оператор наміру:

- C1. $I_i(\alpha \,\&\, \beta) \equiv (I_i\alpha \,\&\, I_i\beta)$;
- C2. $I_i(\alpha \vee \beta) \equiv (I_i\alpha \vee I_i\beta)$.

Наміри не суперечать один одному:

- C3. $\neg I_i(\alpha) \equiv I_i\neg\alpha$;
- C4. $I_i\,\neg(\alpha \,\&\, \beta) \equiv (I_i\,\neg\alpha \vee I_i\,\neg\beta)$;
- C5. $I_i\,\neg(\alpha \vee \beta) \equiv (I_i\,\neg\alpha \,\&\, I_i\,\neg\beta)$;
- C6. $I_i\,\alpha \equiv I_i\neg\,\neg\alpha$;
- C7. $I_i\,(\alpha \to \beta) \to (I_i\,\alpha \to I_i\,\beta)$;
- C8. $I_i\,\alpha \to D_i\alpha$;
- C9. $B_i(I_i\,\alpha) \equiv I_i(\alpha)$.

Очевидно, що запропонований вище набір схем аксіом є несуперечливим, тобто для $\forall\alpha \in F^e$ не виводиться $\alpha \,\&\, \neg\alpha$ (тому що переконання, цілі і наміри кожного конкретного суб'єкта не

суперечать одне одному за визначенням). У той же час запропонована система аксіом у загальному випадку не є повною, тобто $\exists\alpha \in F^e$, для яких не виведені ні α, ні $\neg\alpha$ (це викликано неповнотою знань і переконань конкретних суб'єктів).

2.2. Архітектури агентів

Основні принципи побудови та функціонування ПА називають їх *архітектурою*. Залежно від того, які принципи визначають дії агентів, архітектури поділяються на деліберативні (агенти обирають план дії на основі логічного виведення з наявних в них знань) та реактивні (дії агентів визначаються як реакція на події у зовнішньому середовищі).

На практиці зазвичай застосовують різноманітні комбінації цих архітектур, які називають *гібридними*. Крім того, деякі дослідники виділяють в окремі класи архітектури з певними спільними рисами (приміром, інтерактивні архітектури, архітектури з планувальником дій, архітектури інтелектуальних агентів, інтенсіональні архітектури). Архітектура ПА відображає внутрішню організацію та взаємодію між основними компонентами (рис.2.6).

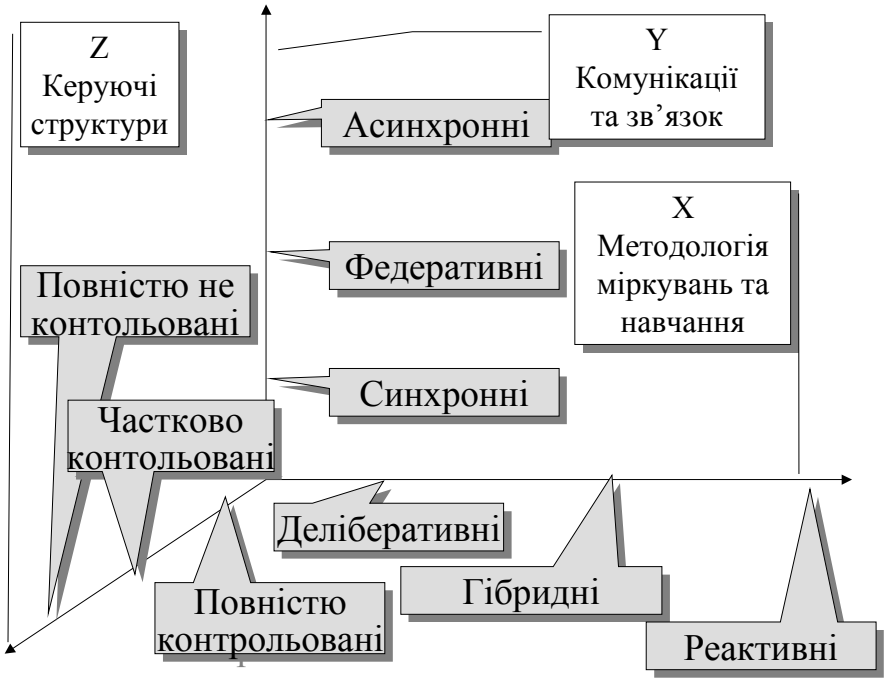


Рис.2.6. Таксономія архітектур мобільних агентів

Приміром, в [265] розглядаються такі характеристики деяких типів архітектур ПА (таб.2.2).

Таблиця 2.2

Характеристики типів архітектур ПА

Архітектура	Подання знань	Модель світу	Вирішувач
Інтелектуальна	Символьне	Числення	Логічний
Реактивна	Автоматне	Граф	Автомат
Гібридна	Змішане	Гібридна	Машина виведення

Деліберативна архітектура ПА містить символьну модель світу, подану у явній формі, за допомогою якої ПА на основі міркувань логічного чи псевдологічного типу приймає рішення про дії, які він здійснює. Такий агент може розглядатися як спеціальний випадок системи, заснованої на знаннях. Використання деліберативної архітектури ПА приводить до кількох принципових проблем:

- обсяг символьної інформації, яку зберігає ПА, пропорційний розміру внутрішнього стану, тому його збільшення призводить до зниження мобільності ПА;
- символьне подання інформації про середовище, на основі якої міркують ПА, має формуватися у режимі реального часу, щоб знайдені агентами рішення були корисними;
- перетворення сценаріїв реального світу на точне та адекватне символьне визначення є нетривіальною задачею.

Реактивні архітектури не використовують централізовану символьну модель світу та не застосовують складні символьні міркування. Така архітектура оперує на низькому рівні абстракції. Невеликий час очікування відповіді забезпечує ефективну взаємодію між агентами з такою архітектурою один з одним та з середовищем. Недолік архітектури – неможливість глибокого аналізу даних від сенсорів.

Реактивна архітектура забезпечує прийняття рішень за значно меншим обсягом інформації про оточуюче середовище та за значно менший час, використовуючи прості емпіричні правила, специфічні для певної ПрО.

Здатний міркувати ПА має явно подану символьну модель світу, у якій рішення (наприклад, про те, яку дію виконувати) продукуються через логічні (або, щонайменше, псевдологічні) міркування, засновані на відповідності зразків і символьних маніпуляцій. Щоб побудувати такого агента, потрібно вирішити дві важливі проблеми:

- *трансдукції:* як перевести реальний світ в коректний адекватний символічний опис за час, протягом якого опис ще буде корисним;

- *подання/міркування*: як у символьному вигляді подати інформацію про сутності і процеси реального світу, і як ПА можуть міркувати над цією інформацією за час, протягом якого результати будуть корисними.

Труднощі з доказом теорем і зі складністю алгоритмів маніпуляції символами існують навіть у досить простих логіках. Алгоритми маніпуляції символами вимагають великих ресурсів. Приміром, логіка першого порядку не розв'язна, а її модальні розширення (включаючи подання переконань, бажань, часу тощо) також не розв'язні. Тому така ідея хоча і дуже приваблива в теорії, але не працює на практиці.

Символьна парадигма ШІ ґрунтується на гіпотезі фізичної символьної системи, сформульованої Невелом і Сімоном [158]. Фізична символьна система – множина фізичних сутностей (символів), що можуть комбінуватися, формуючи структури, і здатні виконувати процеси, що впливають на ці символи згідно в символьному вигляді закодованої множини інструкцій. Гіпотеза фізичної символьної системи стверджує, що така система здатна до загальних інтелектуальних дій.

З іншого боку, архітектури агентів класифікують залежно від типу структури функціональних компонентів ПА та методів організації взаємодії між ними. Як правило, архітектура агента організується у вигляді кількох рівнів.

Можна провести аналогію між архітектурами ПА і комп’ютера (таб.2.3).

Таблиця 2.3

Аналогія між архітектурами ПА і комп’ютера

ПА	Комп’ютер
Сенсори	Пристрої введення
Ефектори	Пристрої виведення
Модуль обробки інформації	Процесор

Тільки найпростіші ПА можуть бути реалізовані за однорівневою схемою. Рівні відображають функції ПА [277], такі, як сприйняття зовнішніх подій і прості реакції на них; координація взаємодії з іншими ПА; відновлення переконань про зовнішній світ; визначення своїх дій на черговому кроці тощо. Найчастіше в архітектурі ПА присутні рівні, що відповідають за такі функції:

- сприйняття і виконання дій;

- реактивну поведінку;
- локальне планування;
- кооперативну поведінку;
- моделювання ПрО;
- формування намірів;
- навчання.

Існує два класи багаторівневих архітектур агентів [249]:

- горизонтальна модульна;
- вертикальна модульна.

Залежно від моделі подання знань використовується у ПА виділяють:

- конекційну архітектуру (з використанням нейронної мережі);
- архітектуру, що базується на правилах.

Залежно від того, які технології ШІ реалізовано в ПА, виділяють архітектуру, що базується, на:

- генетичних алгоритмах;
- знаннях.

Горизонтальна модульна архітектура (рис.2.7) є однією з найпоширеніших. У такій архітектурі функції агента подані окремими модулями у загальному вигляді, без специфікації за задачами.

Між основними блоками існують функціональні зв'язки. Залежно від організації агента вхідна інформація реєструється безпосередньо у БД агента, використовується для навчання агента або для отримання завдань користувача.

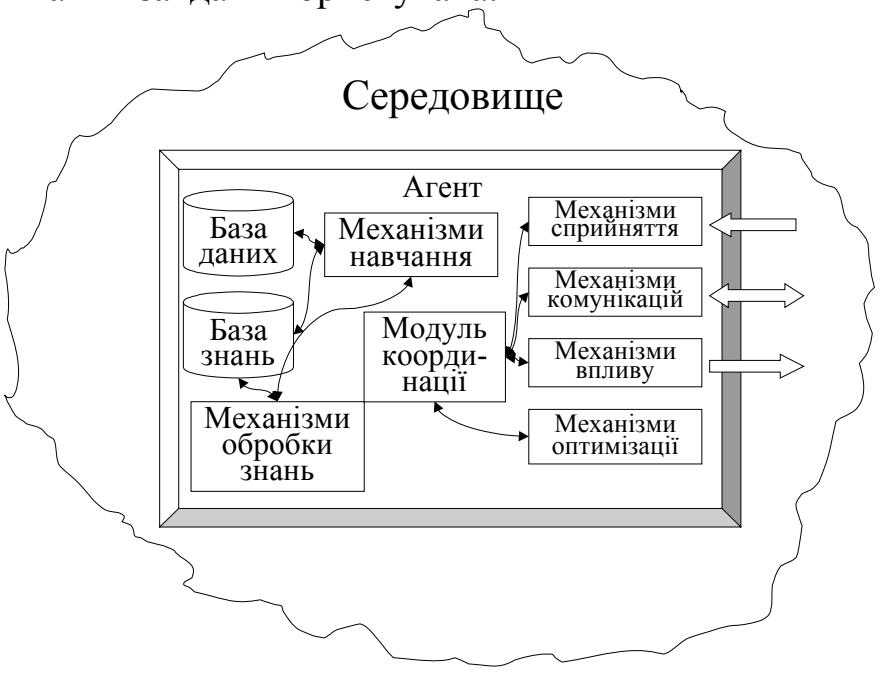


Рис.2.7. Горизонтальна модульна архітектура ПА

Така архітектура характерніша для "розмірковуючих" агентів, а не для реагуючих.

Вертикальна модульна архітектура виділяє модулі агента, які відповідають певним аспектам взаємодії ПА з зовнішнім середовищем. Така архітектура характерніша для реагуючих агентів.

Функціональні зв'язки між модулями є ієрархічними. Модулі виконуються паралельно. Якщо два модулі вступають у конфлікт, розглядаються дані, які поступили від домінантного модуля.

Конекційна архітектура ПА базується на конекційній моделі подання знань, яка використовує багатошарові (multilayer) нейронні мережі. Сигнал, що виходить з одного нейрона, може бути вхідним для іншого. Нейронні мережі використовуються в архітектурі ПА за такими причинами:

- у простих випадках зв'язки встановлюються безпосередньо;
- нейронна мережа може сама встановлювати зв'язки між нейронами через механізм "back-propagation". Різниця між очікуваними та отриманими результатами впливає на встановлення зв'язків між шарами мережі (зворотний зв'язок);
- зв'язки встановлюються за допомогою генетичних алгоритмів.

Архітектура, що базується на правилах (рис.2.8), розглядає ПА як продукційну систему, здатну сприймати середовище та впливати на нього. Сигнали, що поступають від зовнішнього середовища, фіксуються в базі даних ПА. Одночасно з цим працює механізм виведення, який обирає та виконує правила, що відповідають поточному стану БД.

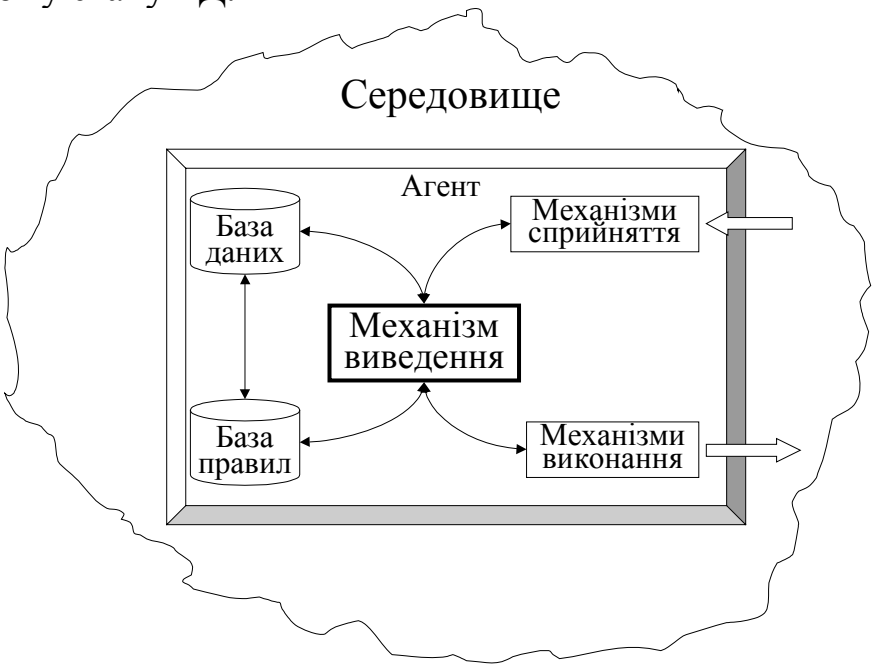


Рис.2.8. Архітектура ПА, що базується на правилах

Архітектура типу "чорна дошка" застосовує поділ на незалежні модулі знань, які взаємодіють один з одним не безпосередньо, а через розподілені дані (рис.2.9). Модулі працюють у просторі, який називають "чорною дошкою". Він містить елементи, які необхідні для вирішення проблем взаємодії. "Чорна дошка" складається з ієрархічно пов’язаних підпросторів: гіпотез, проміжних результатів і різноманітних даних, якими обмінюються модулі. Контрольний механізм керує конфліктами між модулями.

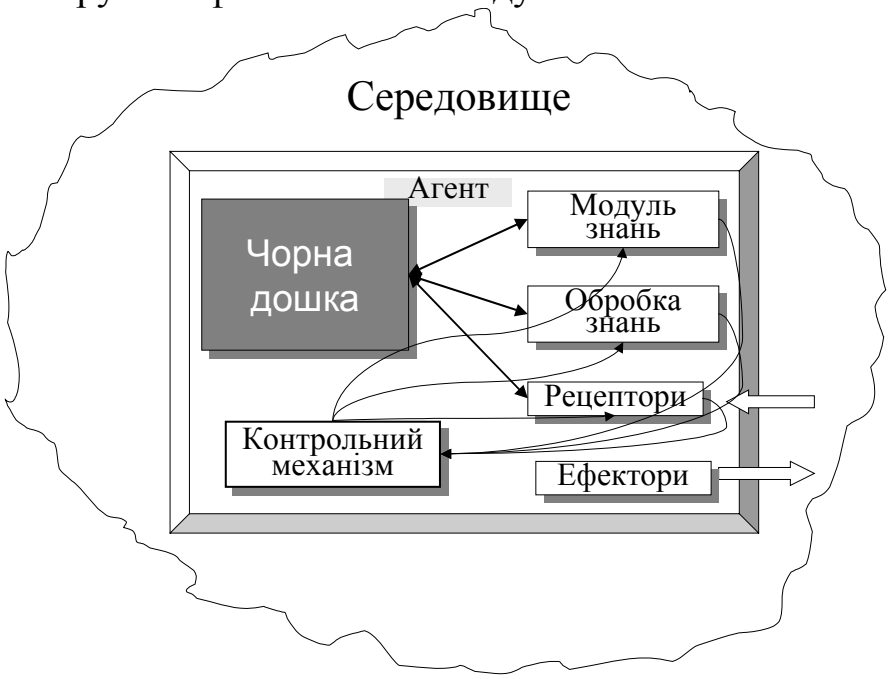


Рис.2.9. Архітектура ПА типу "чорна дошка"

Архітектура на основі генетичного алгоритму (рис.2.10) дозволяє агентам успадковувати властивості двох батьків-агентів.

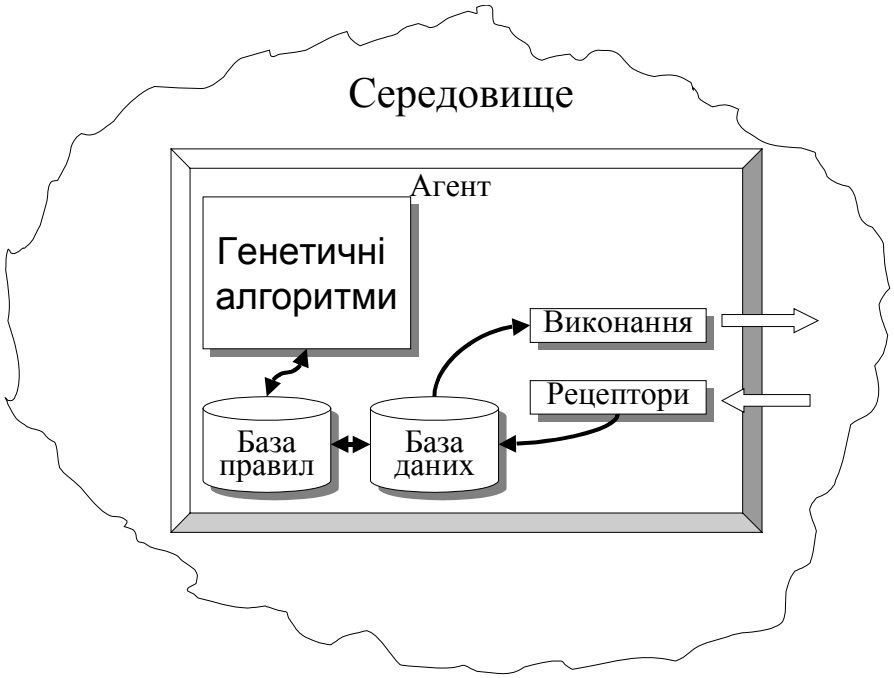


Рис.2.10. Архітектура ПА на базі генетичного алгоритму

Розглянемо найбільш відомі приклади архітектур різних класів, їх преваги, недоліки та галузі застосування.

З початку 1970-х років дослідники в галузі агентних технологій займалися проектуванням *плануючих агентів*. Планування – це автоматичне програмування, тобто проектування послідовності дій, виконання яких має привести до досягнення деякої бажаної мети. Задача планування описується за допомогою початкових станів світу, стану цілі і множині операторів перетворення станів. Задача полягає у визначенні послідовності дій (операторів), що переводять агента з початкового стану у цільовий. Планування можна розглядати як пошук у просторі станів. При цьому подання світу – символьне: стан світу описується множиною тверджень, що дійсні у цьому світі.

Прикладом подібної архітектури є архітектура, у якій реакція ПА на зовнішні події генерується скінченним автоматом. Відома планувальна система *STRIPS* [73] обробляє символьний опис світу і стану бажаної цілі, а також описи дій, що характеризуються перед- і пост-умовами, щоб знайти потрібну послідовність дій. Планувальний алгоритм STRIPS простий, але неефективний на проблемах навіть помірної складності.

Багато праць присвячено розвитку більш ефективних технологій. Наприклад, *IPEM* (Integrated Planning, Execution and Monitoring) заснована на ускладненому нелінійному плануванні. *AUTODRIVE* має плануючих агентів, взаємодіючих у сильно

динамічному оточенні (імітація дорожнього руху), *PHEONIX* включає плануючі ПА, що моделюють гасіння лісових пожеж.

Головними нововведення були ієрархічне і нелінійне планування [203]. Однак теоретичні дослідження показали, що такі технології не можуть використовуватися в системі з обмеженим часом [35]. Ці результати мали великий вплив на наступні дослідження з планувального ШІ і призвели до розробки альтернативних архітектур.

IRMA (Intelligent Resource-bounded Machine Architecture) [24] – архітектура для обмеженого в ресурсах ПА, який обирає свій план дій, ґрунтуючись на явному представленні сприйняття, переконань, бажань і намірів. Ця архітектура має чотири ключові символьні структури даних: бібліотеку планів, явне подання переконань, бажань і намірів. Архітектура системи містить такі модулі:

- *структуру намірів* – упорядковану за часом множину часткових планів деревоподібної структури;
- *розсуджувач* – компонент, в якому реалізовано механізми міркувань про світ;
- *аналізатор можливостей* визначає сукупність дій, які агент може обрати для виконання;
- *аналізатор кінцевих значень* визначає сукупність можливих дій, що наближують агента до виконання власних намірів.

Переконання агента поновлюються за допомогою підсистеми сприйняття.

В [229] розглядається створення прототипу автономного ПА *HOMER*, здатного до планування дій. HOMER моделює підвідний човен, що діє в двовимірному світі, про який ПА має тільки часткові відомості. Агент має лінгвістичні можливості – він одержує інструкції від користувача обмеженою підмножиною слів англійської мови; інструкції можуть включати ускладнені посилання на час. HOMER може планувати виконання цих інструкцій (які звичайно відносяться до пошуку і переміщення різних об’єктів) і потім виконувати свої плани, модифікуючи їх при необхідності в процесі виконання. Агент використовує обмежену епізодичну пам’ять і здатний відповідати на питання про свій минулий досвід.

JAM – це багаторівнева архітектура інтелектуальних агентів, яка базується на ідеях процедурно-міркуючих систем і структурованих семантик [223]. Агент JAM має п’ять компонентів:

- модель світу;
- бібліотеку планів;

- інтерпретатор;
- структуру намірів;
- спостерігач.

Модель світу – БД, в якій зберігаються переконання ПА. Бібліотека планів – сукупність планів ПА, які він може використовувати для досягнення мети. Інтерпретатор дозволяє агенту обирати, які дії йому слід виконати. Структура намірів – внутрішня модель поточних цілей агента та засобів виконання цих цілей. Спостерігач – декларативна процедура, що визначається користувачем.

Зміна моделі світу або виникнення нових цілей змушує ПА шукати плани, які можуть бути застосовані до даної ситуації. Інтерпретатор на основі критерію максимальної корисності обирає з планів один, додає його до своїх намірів та виконує намір, що має найбільшу корисність.

Для кожного плану можна визначити: 1) передумову – умову, яка має виконуватися перед застосуванням плану; 2) контекст – умову, яка має виконуватися як перед, так і під час виконання плану. Така семантика є більш гнучкою порівняно з PRS, що розглянута нижче.

*GRATE** – пошарова архітектура, у якій поведінка ПА описується через ментальні відношення переконання, бажання, наміру й об'єднаних намірів [85]. Структура агента поділяється на дві частини: 1) доменну; 2) взаємодії і керування.

Підсистема взаємодії складається з трьох модулів: керування, оцінки ситуації і взаємодії. Модулі оцінки і взаємодії забезпечують реалізацію моделі спільної відповідальності, яка визначає, як саме агент має діяти (як локально, так і разом з іншими агентами) в процесі спільного рішення певної проблеми.

Дослідники відзначають, що інтелектуальна діяльність людини часто вимагає мало (якщо взагалі вимагає) нових абстрактних міркувань. Значна частина діяльності сутностей, які мають інтелект, (приміром, людей) є рутинною, у тому розумінні, що вона вимагає мало – якщо узагалі вимагає – нових абстрактних міркувань. Велика частина задач, один раз вирішених, можуть бути виконані рутинним шляхом з невеликими варіаціями. Для обробки таких ситуацій агенту найпростіше використовувати механізми швидкого реагування, подібні до безумовних рефлексів, тобто вони виконуються швидко та без особливих розмірковувань.

Саме такий підхід реалізовано в реагуючих архітектурах: більшість дій закодовані в низькорівневих структурах, що потребують тільки періодичного поновлення для охоплення нових типів проблем [27]. При застосуванні такого підходу до архітектури агента включають базу шаблонів поведінки агента, яка складається з наступних:

- множини стандартних реакцій агента (що представлені, наприклад, у вигляді приєднаних процедур);
- правил застосування кожної з визначених процедур;
- правил комбінування процедур у разі необхідності виконання дій, що вміщують деяку послідовність кроків.

Найбільш цікавим є підхід, коли агент має декілька підсистем: одна – розсудлива, що містить символьну модель світу, розробляє план і приймає рішення засобами символьного штучного інтелекту; а друга – здатна реагувати на події навколишнього середовища без проведення складних міркувань. Часто реактивний компонент має деяку перевагу над розсудливим, тому що він може забезпечити швидшу відповідь на важливі події в навколишньому середовищі. ПА має можливість реагувати на події, які потребують миттєвої реакції. Якщо реакція має відбутися протягом визначеного терміну часу, то спочатку розсуджувач здійснює спробу систематизувати інформацію, що надійшла, але якщо цього не вдається зробити у визначений проміжок часу, то застосовується один із відомих шаблонів поведінки ПА.

Альтернативою символьному підходу є *реактивна архітектура,* яка не використовує централізовану символьну модель світу і складні символьні міркування. Інтелектуальна поведінка породжується як результат взаємодії агента з його оточенням, без явних подань та абстрактних міркувань того виду, що пропонує символьний ШІ. Оскільки більшість рішень рутинні, вони можуть бути закодовані в низькорівневих структурах, що потребують тільки періодичне поновлення, можливо, для охоплення нових проблем. Такий підхід використовує система *PENGI*.

Більш складний підхід запропонований у парадигмі ситуативних автоматів [108]. Ця парадигма поєднує кращі елементи як реактивних, так і символьних, декларативних систем. У такому підході є подібність з автоматичним синтезом програм за специфікаціями темпоральної логіки. Агенти задаються в декларативних термінах та потім компілюються. Логіка, використовувана для завдання агента, є модальною логікою знань. Технологія залежить від можливості

завдання конкретної інтерпретації в термінах стану автоматів світам у семантиці можливих світів.

Те, що агент x має інформацію про те, що p має місце в стані світу s, записується як $s \models K(x, p)$, якщо для всіх станів світу, у яких x має те ж значення, що й у s, висловлення p істинно. Агент задається в термінах двох компонентів: сприйняття та дії. Потім використовуються дві програми синтезу ПА: RULER – для задання компонента сприйняття агента; GAPPS – для задання компонента дії.

RULER отримує три вхідних компоненти:

- специфікацію семантики вхідної інформації агентів (приміром, “усякий раз, коли приходить на вхід біт 1, йде дощ”);
- множину статичних фактів (приміром, “усякий раз, як йде дощ, земля мокра”);
- специфікації переходів стану світу (“якщо земля мокра, то вона залишається мокрої, поки не вигляне сонце”).

Програма GAPPS отримує як вхідну інформацію ціль верхнього рівня та множину правил редукції цілей, що кодують інформацію про те, як можуть бути досягнуті цілі, на основі якої генерує програму досягнення цієї цілі.

Мережна архітектура агентів. Маєс описує архітектуру агентів, у якій агент визначається як множина модулів компетенції [128]. Для кожного модуля проектувальник задає перед- і пост-умови (як оператор у STRIPS) і рівень активізації, що задає деяке значення – дійсне число, що показує релевантність модуля конкретної ситуації. Чим вище рівень активізації модуля, тим більше модуль впливає на поведінку агента. Множина модулів компетенції, що задається один раз, складає мережу активізації, у якій модулі з'єднуються один з одним способом, який визначають їх перед- і пост-умови.

Наприклад, якщо модуль a має пост-умову φ, а модуль b має передумову φ, то a і b поєднуються з'єднанням успішності. Інші типи з'єднань включають з'єднання попередника і конфліктні з'єднання. При виконанні агента різні модулі можуть стати більш активними в певній ситуації і можуть бути виконані. Результатом виконання може бути команда на ефектор.

Є очевидна подібність між мережною архітектурою агента й архітектурою нейронних мереж. Можливо, їх ключовою відмінністю є те, що в нейронній мережі важко сказати, що означає вузол, він має значення тільки в контексті мережі взагалі. Оскільки модулі

компетентності визначаються в декларативних термінах, то набагато легше сказати, що вони означають.

Гібридні архітектури. Як видно з наведених вище прикладів, як реактивний, так і деліберативній підхід мають певні переваги та недоліки. Тому доцільно поєднати ці підходи та розробити ПА, який містить дві підсистеми: 1) здатну міркувати, обробляти символьну модель світу, розробляти план і приймати рішення; 2) здатну реагувати на події, що відбуваються в навколишньому середовищі без проведення складних міркувань.

Реактивний компонент має забезпечити швидку відповідь на важливі події в навколишньому середовищі.

Цей вид структури агента веде до ідеї пошарової архітектури, прикладами якої є *PRS* (Procedural Reasoning System) [94] та *INTERRAP* [149]. У такій архітектурі підсистеми керування агентом поєднані в ієрархію з шарами вищих рівнів абстракції. Так, наприклад, самий нижній шар може переводити неопрацьовані дані сенсорів прямо у вихідні дані для ефекторів, тоді як самий верхній шар має справу з довгостроковими цілями. Ключовою проблемою такої архітектури є те, що структура керування взаємодією між шарами вмонтована в підсистему ПА.

Архітектура *PRS* – інтелектуальна система реального часу. Вона надає гнучкі засоби програмування процедур міркування на метарівні. Це дозволяє ПА змінювати поведінку залежно від середовища.

Як і IRMA, PRS базується на BDI-архітектурі і містить явне подання переконань (знань), бажань (цілей) та намірів (поточних планів) ПА. Переконання агента подані через класичну логіку першого порядку. Бажання представлені як засоби поведінки системи, а не як статичне представлення цільових станів.

Бібліотека планів PRS містить плани для різних ПрО, кожний з яких пов'язаний з певною умовою виклику. Активізація плану може бути викликана ціллю або даними. Це дозволяє PRS швидко реагувати на зміни в його оточенні. Множина поточних активних планів в системі складає її наміри. Ці різноманітні структури даних обробляються системним інтерпретатором, що відповідає за поновлення переконань, виклик ОЗ і виконання дій [82].

Архітектура *TOURINGMACHINES* [71] складається з підсистем сприйняття і дій, що взаємодіють безпосередньо з оточенням агента, і трьох шарів керування, вбудованих у відповідну структуру керування, що є посередником між шарами. Кожен шар виконується незалежно

як активний процес. Реагуючий шар генерує потенційні способи дій у відповідь на події, з якими мають справу інші шари. Він реалізований як множина правил "ситуація – дія".

Планувальний шар створює плани та вибирає дії, виконувані в порядку досягнення цілей агента. Цей шар складається з двох компонентів: планувальника і фокусу механізму уваги. Планувальник поєднує генерацію плану і його виконання, використовуючи бібліотеку розроблених планів разом з топологічною картою світу для того, щоб розробити план, що буде виконувати головну ціль агента. Ціль механізму фокусування уваги – обмежити кількість інформації, з якою планувальник має справу, і в такий спосіб поліпшити його ефективність. Він здійснює це завдяки фільтрації нерелевантної інформації, що надходить з навколишнього середовища.

Шар моделювання містить символьне подання когнітивного стану інших сутностей в оточенні агента. Ці моделі обробляються для того, щоб виявити і вирішити конфлікти цілей – ситуації, в яких ПА не може далі досягати своєї мети в результаті несподіваного впливу.

Три шари здатні взаємодіяти між собою (шляхом обміну повідомленнями) і вбудовані в структуру керування. Ціль цієї структури – бути посередником між шарами і, зокрема, вирішувати конфліктні ситуації між різними шарами. Структура керування здійснює це, використовуючи відповідні правила керування.

INTERRAP, як і TOURINGMACHINES, – пошарова архітектура. Кожний послідовний шар представляє більш високий рівень абстракції, ніж той, що нижче його. У INTERRAP ці шари мають подальший поділ на два вертикальних шари: один містить шар БЗ, інший – різні компоненти керування, що взаємодіють з БЗ на відповідних рівнях.

INTERRAP-агент складається з трьох модулів:

- модуля взаємодії зі світом;
- БЗ;
- блоку керування.

На нижньому рівні знаходяться модуль взаємодії зі світом, який забезпечує сенсорну та комунікативну взаємодію, і відповідна БЗ моделі світу. Взаємодія зі світом складається з трьох підсистем, що забезпечують сприйняття середовища, виконання дій, а також засоби фізичної взаємодії з іншими ПА.

Вище компонента інтерфейсу зі світом знаходиться компонент, що обґрунтовує поведінку ПА. Він керує множиною шаблонів поведінки – структур, що містять дії та передумови їх активізації. Дії

можуть бути примітивними або викликати вищі шари для генерації плану.

Вище компонента, що обґрунтовує поведінку, в INTERRRAP знаходиться компонент обґрунтування плану. Він містить планувальник, здатний генерувати плани ПА у відповідь на запити компонента, що обґрунтовує поведінку. БЗ містить бібліотеку планів.

Найвищим рівнем INTERRRAP є компонент взаємодії. Він здатний генерувати об'єднані плани, що задовольняють цілям великої кількості агентів. Ці плани генеруються у відповідь на запити від компонента, що обґрунтовує плани.

Керування в INTERRRAP викликається як ціллю, так і даними. У результаті змін моделі світу різні зразки поведінки можуть бути активізовані, відкинуті або виконані.

Компоненти ментального стану агента INTERRAP, крім сприйняття, поділяються на рівні. Подання поділяються на модель світу з уявленням про середовище; ментальну модель з метарівнями уявлення ПА про себе; соціальну модель з уявленням ПА про інших агентів. Відповідно до рівнів подання визначаються класи ситуацій: поведінкові, що є підмножиною моделі світу ПА; ситуації, що вимагають локального планування, опис якого базується на моделі світу і на ментальній моделі; ситуації, що вимагають спільного планування, опис яких додатково містить частини соціальної моделі ПА. Цілі поділяються на:

- реактивні, які відрізняються від звичайних цілей тим, що вони є короткостроковими і викликаються зовнішніми подіями, що вимагають швидкої реакції або виконання рутинних процедур;
- локальні, які відповідають стандартному поняттю мети в BDI-архітектурі;
- кооперативні, які розподілені між групами агентів.

Функціональна характеристика ПА ґрунтується на функціональних відношеннях між компонентами ментальної моделі ПА. Вони визначають потік керування в агенті, відображення його сприйняття в дії. Цими функціональними відношеннями є:

1) генерація та перегляд уявлення пояснюють відношення між уявленнями ПА і його поточним сприйняттям. Вони мають справу з питанням про те, як сприйняття трансформується в уявлення (генерація уявлень) і як існуючі уявлення змінюються залежно від сприйняття (перегляд уявлення – додавання, скасування або перегляд їх множини відповідно до нового твердження);

2) розпізнавання ситуації – це структурування ситуації відповідно до переконань агента. Воно дозволяє ПА визначити свої активні потреби. В INTERRAP це динамічний процес. Критичні за часом ситуації розпізнаються на основі інформації про модель світу на рівні поведінки. Якщо є більше часу, процес може бути подовжено за допомогою дослідження можливих впливів на локальні цілі агента або взаємодію з цілями інших агентів.

Активізація цілей описує, які з можливих цілей агента є на конкретний момент варіантами вибору в множині ситуацій.

Планування відображає поточні цілі агента в примітиви для їхнього досягнення, тобто в локальні плани, протоколи переговорів і спільні плани. Складання розкладу є процесом об'єднання часткових планів для різних цілей у розклад виконання, що бере до уваги сумісність і (наприклад, тимчасові і попередні) обмеження між різними підпланами. Таким чином, під час складання розкладу агент вирішує, коли і що робити.

Архітектура *COSY* є гібридною BDI-архітектурою, що включає елементи PRS і IRMA. Вона розроблена для MAC DASEDIS [30]. Діяльність ПА буває видів: сприйняття, пізнання, комунікації та керування, кожен з яких моделюється спеціальним компонентом архітектури. COSY містить п'ять головних компонент:

- сенсори;
- маніпулятори;
- комунікаційний;
- когнітивний;
- намірів.

Перші три компонента виконують свої функції безпосередньо: сенсори отримують некомунікативну інформацію від середовища, а маніпулятори дозволяють ПА виконувати некомунікативні дії, тоді як комунікаційний компонент забезпечує обмін повідомленнями.

Компонент намірів містить довгострокові цілі та елементи, що беруть участь у міркуваннях і прийнятті рішень когнітивним компонентом. Когнітивний компонент відповідає за посередництво між намірами ПА і його діями, а також трьома процедурними компонентами: виконання запитів, виконання протоколу та компонентом міркувань, прийняття рішень і реагування. Компонент міркувань, прийняття рішень і реагування є ключовим компонентом у COSY.

Організація МАС за принципом деліберативної архітектури має перевагу з погляду інтелектуальності поведінки та здатності до

логічного виведення. Але створення точної і повної моделі подання світу, процесів і механізмів міркування в ньому представляють певні труднощі. Реактивний підхід дозволяє ефективніше використовувати множину зразків поведінки для реакції агента на певні стимули конкретної ПрО. Застосування цього підходу обмежується необхідністю повного ситуативного визначення всіх можливих активних станів агентів. Поєднання підходів (гібридна архітектура) дозволяє гнучко комбінувати можливості деліберативного та реактивного підходів і поширена на практиці, проте недоліком є відсутність чітких методів проектування.

2.3. Мультиагентні системи

Проблеми, для вирішення яких застосовують агентний підхід, можуть бути досить складними. Дослідження в галузі ШІ показали, що взаємодія і декомпозиція загальної задачі ефективно позначається на результатах виконання цієї задачі. Тому краще створювати окремі автономні блоки такої системи, а потім організовувати їх спільне функціонування.

Термін "*мультиагентні системи*" використовується для позначення ІС, які складаються з множини автономних модулів ПА та мають такі властивості:

- кожен ПА є автономним, мобільним та інтероперабельним;
- ПА, що входять до складу МАС, здатні обмінюватися інформацією для досягнення спільних цілей;
- керування ПА може бути децентралізованим;
- джерела даних і доступ до них децентралізовані;
- робота агентів є асинхронною.

Теорія МАС походить від теорії відкритих систем, розподіленого ШІ і загальної теорії складних систем [355]. Розподілений штучний інтелект (РШІ) пов’язаний з аналізом систем, що складаються з окремих незалежних об'єктів, які взаємодіють один з одним, та механізмів їх координації. МАС теж є предметом розгляду РШІ. У цьому випадку незалежними об’єктами є ПА. Для вивчення поведінки МАС використовують методи таких наукових дисциплін:

- *розподілений ШІ* (теорія розподілених систем, теорії прийняття рішень) [241], що займається найбільш загальними аспектами колективної поведінки агентів;
- *теорія ігор* [152], яка використовується для дослідження ситуацій, аналогічних до кооперативних ігор, стратегій ведення переговорів;

- *теорія колективної поведінки автоматів* [257], яка досліджує колективну поведінку великих груп автоматів з примітивними функціями, спроможних навчатися за допомогою системи штрафів і заохочень;
- біологічні, економічні та соціальні моделі.

У розробці МАС використовуються також результати з інших галузей теоретичних досліджень (рис.2.11).

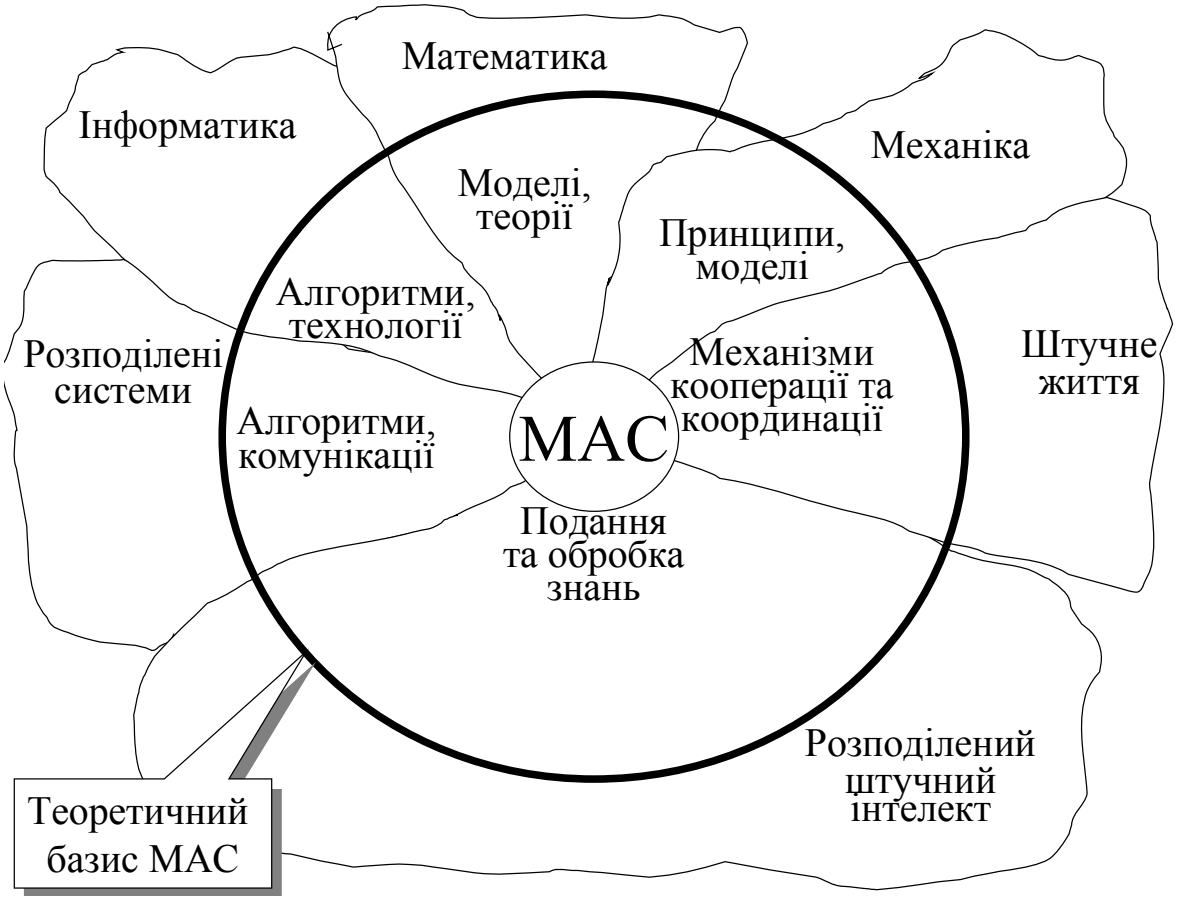


Рис.2.11. Основні ПрО, що використовуються у розробці МАС

МАС – співтовариство ПА, які пов'язані один з одним цілями та ресурсами. ПА, що входять до складу МАС, здатні взаємодіяти для того, щоб обмінюватися послугами, які потрібні їм для досягнення цілей, поставлених перед ними користувачами. Агент, не здатний самостійно вирішити задачу, поставлену перед ним користувачем, звертається до інших агентів, які можуть надати йому відповідні послуги.

МАС поширені в багатьох ПрО: керуванні виробничими процесами і промисловими підприємствами; плануванні рухом транспорту (повітряного, залізничного, автомобільного); аналізі та пошуку інформації; навчанні; бізнесі тощо. Одне з перших

застосувань МАС – DVMT (Distributed Vehicle Monitoring), в якому географічно розподілені ПА контролюють транспортні засоби. В галузі керування виробництвом першим застосуванням МАС – YAMS (YET ANOTHER MANUFACTURING SYSTEM).

МАС складається з множини ПА, які [239]:

- взаємодіють шляхом комунікацій;
- здатні діяти у певному середовищі;
- мають певні сфери впливу, які можуть перетинатися або співпадати.

Можна розглядати МАС як слабкопов'язану мережу обчислювальних пристроїв, взаємодіючих для вирішення проблеми, що знаходиться поза індивідуальними можливостями або знаннями кожного з цих пристроїв [215].

Координація – одна з центральних проблем МАС. Наступні фактори визначають потребу в координації МАС [161]:

- запобігання хаосу в децентралізованій МАС;
- наявність глобальних обмежень, яким має задовольняти МАС для успішного виконання завдання;
- розподіленість знань, ресурсів та інформації, які використовують ПА, що входять до складу МАС;
- залежність між цілями та діями ПА;
- забезпечення ефективності функціонування МАС.

Функціонування МАС пов'язане з кооперацією та конкуренцією агентів в процесі колективного вирішення задач. Агент, що не може вирішити власну задачу самостійно, має взаємодіяти з іншими ПА. При цьому агенти можуть будувати плани спільних дій, ґрунтуючись не тільки на власних можливостях, але і аналізувати плани і наміри інших ПА (використовуючи різні комбінації інтенсіональних відношень). Кооперативні дії ПА – важлива перевага МАС. При такій організації роботи група ПА проявляє новий, ефективніший тип поведінки.

Мотивація використання МАС базується на їх властивостях:

- здатність вирішувати проблеми, складні для одного централізованого ПА через обмеженість ресурсів;
- можливість взаємодії та інтероперабельності з різноманітними застосунками (ЕС, СППР тощо);
- вирішення дійсно розподілених проблем (приміром, керування рухом транспорту);
- знаходження рішень на основі розподілених ІР;

- вирішення проблем з розподіленою експертизою (приміром, медичний догляд);
- підвищення швидкості та надійності;
- здатність збільшувати кількість обчислювальних пристроїв (процесорів), що використовуються для виконання задачі;
- можливість опрацювання нечітких та неповних даних і знань.

Знання й уміння МАС здобуваються від великої кількості відносно простих ПА, що поєднуються разом деякою архітектурою. В процесі колективної роботи агенти мають будувати плани дій, ґрунтуючись не тільки на своїх можливостях, але й враховуючи плани і наміри інших агентів (на основі своїх знань та переконань щодо цих планів та намірів).

Причини взаємодії ПА:

- наявність сумісних цілей;
- нестача індивідуальних ресурсів ПА для досягнення цілей;
- нездатність ПА самостійно вирішити задачу;
- наявність взаємних зобов'язань.

Перехід до відкритих МАС надає можливість переходу на нову якість функціонування системи в силу того, що система (група агентів) більше, ніж сума властивостей її членів (агентів) [348].

Системи, які складаються з досить простих програм, кожна з яких переслідує тільки свої примітивні цілі, у цілому здатні вирішувати дуже складні задачі [278]. Однак моделювання колективної поведінки приводить до необхідності рішення багатьох проблем:

- формування спільних планів дій;
- врахування інтересів інших агентів;
- синхронізації спільних дій;
- вирішення конфліктів між цілями різних агентів;
- конкуренції за спільні ресурси;
- організації переговорів про спільні дії;
- розпізнавання необхідності кооперації;
- вибір партнерів;
- декомпозиції задач;
- поділу обов'язків тощо.

Багато робіт в сфері ШІ присвячені специфікації протоколів взаємодії між ПА. Дослідження переговорів у МАС засновані на ідеї контрактних мереж. Відповідно до цієї парадигми ПА, який бажає отримати певну послугу, запрошує на переговори інших ПА і вибирає послуги, які найбільше відповідають його потребі. У складніших

моделях ПА тільки запитує пропозиції, а інші спеціалізовані агенти оцінюють варіанти відповідей та обирають найкращу.

Інший вид взаємодії між ПА – формування коаліції для спільного виконання своїх намірів. Механізми формування коаліції, де агенти динамічно поєднуються з іншими агентами, інтенсивно вивчаються багатьма дослідниками [242].

Існує багато моделей кооперації агентів. Приміром, модель CPS (Cooperative Problem Solving) [105, 242] призначена для встановлення взаємодії між ПА, побудованих на основі BDI-архітектури. Ментальні поняття формалізуються за допомогою операторів темпоральної логіки для визначення таких понять, як потенціал кооперації, групові дії, досяжність мети агента тощо.

Найпростіший метод координації МАС – *організаційне структурування* – полягає у чітко визначених і довгострокових відношеннях між ПА. При цьому використовують ієрархічні структури *master-slave* або *client-server*. Організаційне структурування передбачає, що принаймні один ПА має глобальне уявлення про все співтовариства, однак для багатьох ПрО це не реально.

Цей метод використовується в двох варіантах:

- головний ПА планує і розподіляє завдання між підлеглими ПА, які, на відміну від головного, мають часткову автономність;
- для координації використовується дошка оголошень для обміну інформацією між ПА, операції з якою (зчитування й запис) визначає головний ПА.

Вузьким місцем МАС, які базуються на технології дошки без прямої взаємодії ПА, є збільшення кількості агентів у співтоваристві та необхідність спільного представлення про дошку. Тому більшість систем, заснованих на цій технології, використовують невеликі гомогенні ПА (наприклад, DVMT).

Інший варіант координації – *укладення контракту* – припускає децентралізовану структуру. ПА мають дві ролі:

- менеджер поділяє задачу на підзадачі і шукає виконавця, щоб виконати їх;
- виконавець реалізує свою підзадачу.

Виконавець може рекурсивно стати менеджером і розбити свою підзадачу на ще дрібніші задачі і доручити їх іншим ПА.

Пошук виконавця складається з таких етапів:

- менеджер повідомляє задачу;
- виконавці оцінюють цю задачу щодо можливості її виконати;

- менеджер отримує таблицю виконавців, оцінює отримані пропозиції, обирає виконавця і доручає йому виконання певної задачі;
- виконавець виконує отриману задачу та повідомляє менеджера про отримані результати.

Така модель координації забезпечує розподіл задачі і засоби для самоорганізації співтовариства агентів. Її доцільно застосовувати у таких ситуаціях:

- задача має чіткий ієрархічний характер;
- задачу можна поділити на великі підзадачі;
- взаємозв'язок між підзадачами – відносно невеликий.

Переваги цього підходу координації полягають у динамічному розподілі задачі завдяки пошуку оптимального виконавця; збалансованому завантаженні ПА і надійному механізмі для розподіленого керування.

Існує два типи планування діяльності МАС:

- централізоване;
- розподілене.

У *централізованому* плануванні діяльності МАС є координаційний ПА, що отримує індивідуальні плани від інших, аналізує їх, щоб знайти потенційні протиріччя та конфлікти у взаємодії ПА. Потім координаційний ПА намагається змінити ці індивідуальні плани і поєднує їх у план МАС, у якому усунені суперечливі взаємодії та додані команди зв'язку, що синхронізують взаємодію ПА у потенційно можливих конфліктах.

У *розподіленому* плануванні діяльності МАС кожен ПА отримує моделі планів інших ПА, що входять до складу МАС. ПА взаємодіють, щоб побудувати модифікації своїх індивідуальних планів, які не конфліктують з планами інших ПА. Координація в розподіленому плануванні діяльності МАС набагато складніша, ніж у централізованій випадку, тому що жоден ПА не володіє глобальним уявленням про всю розподілену систему.

Переговори – один з найскладніших методів координації, який використовує методи ШІ, логіку тощо. У динамічному співтоваристві агенти самостійно розподіляють роботи між собою. Часто одну і ту ж роботу в співтоваристві можуть виконати кілька ПА. Агенти на момент розподілу робіт знаходяться в різних станах (приміром, мають різний ступінь завантаженості). Для ефективного розподілу робіт між ПА вони мають взаємодіяти один з одним – вести переговори, у

процесі яких виявити оптимального виконавця для кожної роботи [288].

Процес переговорів складається з таких компонентів:

- протоколу переговорів – набору правил, за якими відбувається взаємодія агентів (учасники переговорів, їх типи, стани переговорів, правила, за якими змінюються стани переговорів, можливі дії учасників тощо);
- об’єкта переговорів – діапазону проблем, відносно яких потрібно досягти згоди;
- моделі ухвалення рішення ПА – апарат ухвалення рішення, що використовують учасники відповідно до протоколу переговорів.

У процесі переговорів агент-ініціатор переговорів шукає потенційного агента-виконавця для виконання деякої роботи. З огляду на раціональність агентів розумним було б припустити, що об'єктом переговорів агентів на стадії розподілу робіт є розмір винагороди, яку одержує агент-виконавець за ці роботи.

У процесі формування кооперативного рішення в CPS-моделі виділяють чотири етапи:

1. *Визначення потреби у кооперації* (приміром, агент має певну мету, але переконаний у тому, що не може досягти самостійно).

2. *Створення групи агентів.* При успішному завершенні цього етапу створюється група ПА зі спільними зобов'язаннями, пов’язаними з виконанням колективних дій.

3. *Формування спільного плану дій*. ПА ведуть переговори для формування плану дій, який за їхніми переконаннями має привести до реалізації цілей кожного з них.

4. *Спільні дії*. Агенти виконують дії відповідно до плану, який вони прийняли на попередньому етапі, виконуючи прийняті на себе зобов'язанням.

Кожний з підходів до координації МАС має певні переваги і недоліки та може застосовуватися тільки для деяких специфічних областей. Універсального методу координації, що ідеально підходить для будь-якої реальної задачі, не існує.

Отже, при побудові МАС, що вирішує реальну задачу, як правило, необхідно використовувати комбінацію описаних вище підходів. При моделюванні діяльності віртуальних і реальних підприємств переважно застосовують організаційне структурування, яке задовільно відображає ієрархічну структуру підприємства. Крім того, це один з найпростіших підходів до координації. Інші методи координації мають серйозні недоліки при вирішенні задач подібного

роду: переговори – через складність реалізації; планування МАС придатне лише для специфічних задач, таких, як планування авіарейсів. Укладення контракту при моделюванні віртуальних підприємств не надає переваг порівняно з організаційним структуруванням, тому що мережа контрактів фактично визначена до початку роботи і тому немає необхідності у пошуку виконавця.

Для МАС характерні децентралізованість даних, асинхронність обчислень і наявність засобів комунікації. Впровадження МАС дозволяє вирішувати проблеми, що є надто складними для окремого ПА або пов'язані з обмеженими ресурсами, та забезпечує збільшення ефективності системи через паралельні обчислення та повторне використання ПА в різних співтовариствах агентів. Для успішного функціонування вони потребують великих обсягів знань, які мають постійно оновлюватися та перевірятися. Це досягається шляхом обміну знаннями між ПА.

Проте використання МАС пов’язане не тільки з перевагами, але й з новими проблемами [277]:

- формування спільних планів дій;
- врахування інтересів інших агентів;
- синхронізація спільних дій;
- вирішення конфліктуючих цілей;
- наявність конкуренції за спільні ресурси;
- організація переговорів між агентами;
- вибір партнерів для кооперації;
- декомпозиція задач;
- розробка правил колективної поведінки.

Потрібно мати модель середовища, в якому діють ПА. Фактичний результат дій ПА залежить від комбінації дій усіх ПА.

МАС можуть бути як централізованими, так і децентралізованими. У централізованих МАС, створених для вирішення агентами спільних задач, конфлікти між цілями агентів не виникають (або ж існують централізовані механізми вирішення таких протиріч). Приміром, у інформаційно-пошуковій МАС, що обслуговує групу користувачів, пріоритети користувачів визначають порядок їх обслуговування. Проте такі МАС значно важче адаптувати до нових потреб користувачів або нових завдань.

Основні типи конфліктів у МАС [277]:

- в системі переконань ПА, що можуть виникнути внаслідок отримання від іншого ПА інформації, що є хибною або суперечить його переконанням;

- обумовлені неповнотою моделі середовища і моделей інших ПА;
- внаслідок конкуренції ПА за спільні ресурси або через суперечливість їх цілей.

Вирішення конфлікту – це зняття логічного протиріччя шляхом відкидання однієї або обох альтернатив відповідно до певного критерію. Існує багато різних механізмів вирішення конфліктів: централізовані, імовірнісні, відповідно до певних правил (на основі пріоритетів переконань, рівнів компетентності ПА) тощо.

Значна частина МАС, створених на сьогодні, складається з ПА, які не є повністю автономними (тобто здатними вибирати для себе цілі і вирішувати, яким чином досягати ці цілі та які саме дії виконувати для цього). Набір ПА розробляється цілісно, і проблема взаємодій між агентами вирішується в процесі їхньої розробки. Такі проблеми, як конфлікти або недостатність ресурсів, узагалі не розглядаються. Агенти, що входять до складу такої МАС, мають обмежену автономію: їхня роль у процесі рішення загальної проблеми звичайно заздалегідь визначається розробником системи, але агенти вільні у виборі способів досягнення своїх цілей.

Намагання ПА бути корисними для усіх навіть за рахунок власних інтересів є значним недоліком з точки зору їхньої автономності, тому що вони не враховують при виборі цілі витрачених ними зусиль на виконання потреб інших агентів. Хоча такий підхід надає багато корисних можливостей, він не дозволяє використовувати весь потенціал агентної парадигми. Зараз значна кількість розробників ПЗ використовують більш конструктивний підхід до розробки МАС, у якому центральним об’єктом аналізу є окремі ПА, а не система в цілому. Цей підхід більше відповідає специфіці відкритих і розподілених застосувань.

Характер зв’язків у МАС визначають два основні фактори:

- *тип і ступінь взаємодії* між ПА. Приміром, ПА, який має відповідні повноваження, може заборонити певні дії інших агентів (агент захисту інформації може відмовити у доступі до БД агентам, які не повідомляють пароль);
- *стратегія прийняття рішень* різними агентами. У більшості випадків, агенти намагаються максимізувати власну користь, а успіх кожного агента є єдиною мірою загальної оцінки ефективності МАС.

Для опису МАС недостатньо використовувати таку абстракцію, як рівень знань. Потрібний вищий ступінь абстракції – соціальний рівень, на якому розглядаються такі явища, як кооперація,

координація, конфлікти і змагання. Саме на цьому рівні описуються структури і механізми, що мають бути присутніми у МАС для забезпечення бажаного типу поведінки. Щоб розглядати МАС на цьому рівні, потрібно визначити наступне:

- систему (сутність, що описується на цьому рівні) – МАС;
- компоненти (примітивні елементи, з яких будується система) – окремі ПА;
- закони композиції (правила, що визначають, як взаємодіють компоненти системи);
- закони поведінки (правила, що визначають, як поведінка системи залежить від поведінки її компонентів);
- проміжний рівень (посередники – елементи системи, призначені для забезпечення необхідної поведінки).

Ці атрибути забезпечують основні точки зору – структурну, поведінкову і функціональну – завдяки яким може бути описана система (таб.2.4).

Таблиця 2.4.

Визначення атрибутів рівня знань і соціального рівня

Атрибути	Рівень знань	Соціальний рівень
Система	Агент	Відповідальне співтовариство
Компоненти	Цілі та дії	Агенти, оточення, способи і цілі взаємодії
Закони композиції	Відсутні	Різноманітні
Закони поведінки	Принцип раціональності	Принцип соціальної раціональності
Проміжний рівень	Знання	Знайомство, вплив, права й обов'язки

Існує багато різноманітних типів МАС, але можна виділити характеристики, яким задовольняють деякі з цих систем:

- агенти, що входять до складу МАС, автономні;
- агенти поєднують свої індивідуальні потреби з потребами всієї системи.

МАС, що відповідає цим умовам, називають *відповідальним співтовариством.* Його компонентами є:

- *елементи*, призначені для рішення певних проблем;
- *середовище*, у якому знаходяться ці елементи;
- *способи взаємодії* між елементами;

- *цілі* – мотиваційна сила, під впливом якої елементи вирішують проблеми.

Разом ці компоненти характеризують систему.

Базовим елементом є соціально відповідальний ПА. Крім того, елементи можуть складатися з наборів таких агентів або з інших елементів. Таке рекурсивне визначення має наступні переваги: довільна складна структура співтовариства може бути подана однаковими засобами, а опис співтовариства можна перенести на його елементи. Таким чином, елемент співтовариства може бути подано на рівні знань. Відмінність між відповідальним ПА і базовим елементом полягає у рівнях подання. Якщо сутність подано як елемент, тоді її розглядають на соціальному рівні, а якщо її подано як агент, то її розглядають через поняття рівня знань. В обох випадках вона має однакові загальні властивості і є тією же самою сутністю.

Середовище – це контекст, у якому елементи вирішують проблеми (приміром, Інтернет або корпоративна мережа). У більшості МАС існує двобічна взаємодія між ПА і середовищем. Таким чином, ПА можуть виконувати дії, що змінюють їхнє середовище, а зміни в середовищі можуть впливати на дії агентів.

Елементи співтовариства вирішують проблеми для того, щоб досягти певних цілей. Цілі можуть бути *чіткими* (приміром, зустрітися з іншим ПА у конкретно визначений час) або *нечіткими* (приміром, досягти успіху). Вони можуть бути *подані явно* (як у BDI-архітектурі) або *неявно* (як у реагуючих архітектурах). Цілі можуть бути *індивідуальними* і належати одному елементу або бути *колективними* і належати кільком елементам або навіть усьому співтовариству. Цілі елементів можуть виконуватися трьома способами:

- діями, які елементи виконують самі;
- діями, які виконують інші елементи;
- комбінацією першого і другого способів.

Закони композиційних груп – це методика структурування елементів співтовариства. Ця структура містить інформацію про взаємні залежності між існуючими агентами й у такий спосіб визначає соціальну поведінку. Не існує єдиної найкращої організації МАС. Який тип організації є ефективнішим, залежить від функцій системи (іноді відповідальне співтовариство в процесі свого функціонування може навіть змінити тип організації для кращого рішення проблеми).

Законом, що визначає поведінку МАС, є *принцип соціальної раціональності* (на рівні знань йому відповідає принцип раціональності). Розглянемо детальніше таку МАС:

- S – відповідальне співтовариство;
- M – відповідальний елемент S, $M \in S$;
- A(M) – набір дій, що може виконати M;
- дія a – елемент A(M), $a \in A(M)$.

Кожна дія a, яку виконує M, приносить певну користь усьому відповідальному співтовариству S. Визначимо результуючу користь B(M,a) як суму функцій b(M,a,N), що визначають користь цієї дії для кожного елемента відповідального співтовариства $N, N \in S$. $B(M,a) = \sum_{N \in S} b(M,a,N)$.

Користь визначається як різниця між вигодою та витратами. Вигода – це те, що здобувається в результаті виконання дії. Витрати – вартість виконання дії. p(M,a,N) – вигода для N, якщо M виконує дію a. P(M,a) – вигода для S, якщо M виконує дію a. $P(M,a) = \sum_{N \in S} p(M,a,N)$.

v(M,a,N) – витрати для N, якщо M виконує дію a. V(M,a) – витрати для S, якщо M виконує дію a. $V(M,a) = \sum_{N \in S} v(M,a,N)$.

Щоб перебрати всі можливі варіанти тільки один раз, передбачається, що всі елементи складаються з наборів ПА, які не перетинаються, $\forall M, N \in S, M \neq N, M \cap N = \varnothing$, а їх об'єднання покриває всіх агентів відповідального співтовариства $\bigcup_{M \in S} M \equiv S$.

$B(M,a) = P(M,a) - V(M,a)$.

Якщо певна дія M не приносить користь усьому відповідальному співтовариству, то така дія не буде виконуватися. Якщо ж певна дія a виконується, то завжди $B(M,a) > 0$.

Визначати, які дії ПА є соціально корисними, можна за різними принципами. Наприклад, агент може виконувати дію, якщо вигода для нього більше, ніж витрати на його виконання, і не враховувати вигоду для усього відповідального співтовариства, або виконувати тільки ті дії, індивідуальна і соціальна вигода від яких вище витрат (на практиці звичайно задається функція $\varphi(P(M,a), V(M,a), p(M,a), v(M,a))$ від цих параметрів, що має два значення – "*виконувати*" і "*не виконувати*"). Залежно від вигляду цієї функції можна виділити різні типи агентів: індивідуально раціональні, доброзичливі, егоїстичні тощо.

2.4. Мови комунікації агентів

ПА можна розглядати як застосунки, для яких здатність до комунікації з іншими застосунками і до пошуку знань має першочергову важливість. Компоненти ПА можна поділити на такі: подання, комунікацій і ті, що не відносяться безпосередньо до розподіленої взаємодії.

Для рішення проблеми комунікацій між інтелектуальними ПА необхідна спільна мова, тобто спільні синтаксис, семантика й прагматика. Зараз не існує загальноприйнятої мови подання інформації і запитів. Такі мови, як KIF і SQL, досить поширені, але цього недостатньо, щоб прийняти їх як стандарт.

Агенти мають взаємодіяти на рівні знань і тому не можуть задовільно обслуговуватися звичайними мовами та протоколами, що призначені для розподілених обчислень і звичайно фокусуються на процесах, а не на програмах або коаліціях ПА. Мова комунікації агентів має бути досить потужною для підтримки взаємодії між програмами високого рівня. Мова комунікації – не тільки протокол.

Протокол взаємодії – це високорівнева стратегія, що здійснюється агентом і взаємодіє з іншими агентами. Такий протокол може знаходитися в діапазоні від схеми переговорів і протоколів теорії ігор до простих протоколів типу "*щораз, коли я щось не знаю, я знаходжу кого-небудь, хто знає, і я запитую його*". Вибір транспортного протоколу залежить від специфіки його застосування.

Протокол, що використовується в контексті мови комунікацій, може мати одне з таких значень:

- транспортний протокол (HTTP, SMTP, FTP);
- високорівнева структура взаємодій (переговори, протоколи теорії ігор, планування);
- обмеження (constraints) на можливі зміни в комунікаційних примітивах.

Мова комунікацій може використовувати протоколи першого типу як транспортний механізм, другого – як спосіб їх використання, і третього – як частину визначення.

Можна сформулювати такі вимоги до мови взаємодії агентів:

- *форма*: декларативна, синтаксично проста, зрозуміла;
- *зміст*: відрізняє мови опису комунікаційних актів від мов передачі вмісту повідомлень;
- *семантика*: подає бажані властивості;
- *реалізація*: проста, сумісна з існуючим ПЗ;

- *робота в мережі*: враховує основні аспекти сучасних ІТ і має незалежний транспортний механізм ;
- *середовище*: підтримує гетерогенність і динамізм;
- *надійність*: забезпечує надійність і безпеку взаємодії агентів.

Ці вимоги можуть конфліктувати одна з одною. Приміром, проста для розуміння людиною мова може бути не такою стислою, як можливо. Вибір вдалого співвідношення усіх вимог являє собою для розробника мови нетривіальну задачу.

Крім того, мова комунікацій має забезпечувати надійність**,** конфіденційність і безпеку взаємодії агентів на основі механізму ідентифікації агентів і виявлення помилок.

Для конструктивної й інтелектуальної взаємодії між ПА необхідна не тільки загальна мова, але й загальне розуміння термінів, що використовуються у конкретному контексті.

Таким чином, ПА взаємодіють в трьох напрямках:

- спільної мови;
- спільного розуміння знань, якими вони обмінюються;
- здатності обмінюватися знаннями за допомогою спільної мови.

Петрі [176], проаналізувавши різні спроби відрізняти агентів від інших типів ПЗ, першим відзначив складність задовільного їх визначення інтелектуальності й автономії.

Він виділив особливий тип ПА – *typed-message agents*, специфічною властивістю яких є здатність спілкуватися за допомогою високорівневих протоколів повідомлень типу *KQML* (Knowledge Query Manipulation Language) [75].

Зараз активну діяльність щодо стандартизації агентів підтримують такі комітети: FIPA (Foundation of Intelligent Physical Agents), Agent Society, OMG, W3C, Agent X, Knowledge Sharing Effort. FIPA пропонує дві специфікації для різних аспектів ПА:

- мову комунікацій ПА ACL (Agent Communication Language);
- керування ПА (Agent Management).

Структура повідомлення мовою ACL подана у таб.2.5.

Таблиця 2.5

Структура повідомлення ACL

Елемент	Опис
Performative	Перформатив
Sender	Відправник повідомлення
Receiver	Одержувач повідомлення
Reply-to	Адреси ПА, яким відправляти відповідь

Content	Вміст повідомлення
Language	Мова кодування повідомлення
Encoding	Кодування вмісту повідомлення
Ontology	Онтологія, що використовується для інтерпретації повідомлення
Protocol	Протокол взаємодії ПА
Conversation-id	Ідентифікатор повідомлень
Reply-with	Рядок, що ідентифікує повідомлення
In-reply-to	Рядок, що ідентифікує повідомлення, яке відповідає параметру reply-with при відповіді на повідомлення
Reply-by	Час, до якого необхідно отримати відповідь

На сьогодні найбільше поширення набула еталонна модель керування агентами, що базується на специфікаціях FIPA. FIPA пропонує термінологію та задає нормативну структуру, в якій розглядається функціонування ПА [80].

Абстрактну архітектуру агента .можна подати через сервіси:

- керування платформою агента APMS (Agent Platform Management Services);
- передачі повідомлень агентів AMTS (Agent Message Transport Services);
- каталогу агента ADS (Agent Directory Services).

Спеціальні механізми, що забезпечують керування ПА – це:

- посередник каталогу DF (Directory Facilitator):
- система керування агентами AMS (Agent Management Systems);
- канал комунікацій агентів ACC (Agent Comunication Channel);
- механізм передачі міжнародної платформи IPMT (Internal Platform Message Transport).

Агент – основний актор агентної платформи, який поєднує одну або більше сервісних властивостей в уніфіковану та інтегровану модель виконання, яка може включати доступ до зовнішнього ПЗ, користувачів і засобів комунікацій. Агент може мати певні можливості для розподілу ресурсів. ПА обов’язково має одного або кількох власників (приміром, членів якоїсь організації або людину-користувача). Глобальний унікальний ідентифікатор GUID (Globally Unique Identifier), тобто ім’я агента, однозначно ідентифікує агента в усіх доменах FIPA в просторі агентів. Агент може бути зареєстрований як множина адрес, за якими з ним можна контактувати.

Посередник каталогів DF (Directory Facilitator) забезпечує сервіси "жовтих сторінок". DF – обов’язковий нормативний ПА. Інші агенти можуть реєструвати свої сервіси через DF або звертатися до DF, щоб знайти сервіси, запропоновані іншими агентами. Він зберігає визначення агентів і сервісів, що вони пропонують, а також зв'язки між глобальними унікальними іменами агентів GUID (globally unique agent names) і локальними транспортними адресами.

Система керування агентами AMS (Agent Management System) також є обов’язковим компонентом агентної платформи. Це агент, який здійснює контролююче керування через доступ та використання агентної платформи. В кожній агентній платформі існує тільки один AMS. AMS здійснює підтримку каталогу логічних імен агентів та пов’язаних з ними транспортних адрес для агентної платформи. Крім того, AMS пропонує сервіси "білих сторінок" іншим агентам.

Кожний агент має доступ хоча б до одного *каналу комунікації агентів* ACC (Agent Communication Channel). ACC – це метод комунікації між агентами різних агентних платформ, заданий за замовчуванням. ACC використовує інформацію системи керування агентами для маршрутизації повідомлень між агентами усередині платформи і до агентів інших платформ. Сервіс маршрутизації повідомлень, який пропонує ACC, має бути надійним та систематизованим. Він відповідає FIPA Baseline Protocol та ACC.

Інфраструктура середовища, у якому функціонує ПА, називається платформою агента. *Агентна платформа* AP (Agent Platform) забезпечує фізичну інфраструктуру, в якій агенти можуть розгортатися. Агенти мають бути зареєстровані в платформі для взаємодії з іншими агентами цієї платформи або інших платформ. AP містить операційну систему, ПЗ підтримки агентів, компоненти керування агентів FIPA (DF, AMS, ACC), Internal Platform Message Transport та агентів.

Локальна агентна платформа – агентна платформа, яка визначає місце призначення для повідомлень, які направляються агенту. Для визначення агентів використовують мову ADL (Agent Definition Language).

Внутрішня структура агентної платформи не є предметом стандартизації FIPA та визначається розробниками AP. Агентні платформи та агенти, безпосередньо розроблені для цих платформ або мігрувавши до них, можуть використовувати будь-який власний метод для комунікації між собою.

Слід відмітити, що концепція агентної платформи не означає, що агенті однієї АР мають бути розташовані на одному хості. FIPA розглядає тільки те, як здійснюються комунікації між агентами усередині АР та поза її межами.

Транспорт повідомлень внутрішньої платформи IPMT (Internal Platform Message Transport) – власні засоби для обміну повідомленнями усередині агентної платформи, що знаходиться поза межами FIPA.

Домен агента – це логічне групування агентів, яке визначається членством у каталозі, який підтримує DF. Кожний домен має один та тільки один DF, який забезпечує уніфікований, повний та послідовний опис домену. Агент може бути присутнім в одному або у кількох доменах. Частиною життєвого циклу агента є його реєстрація в DF, яка забезпечує присутність агента у домені. Агентна платформа може підтримувати більше ніж один домен. *Універсум агентів* (Agent Universe) – це набір усіх доменів агентів.

Ця логічна модель використовується для опису створення, реєстрації, місцезнаходження, міграції та видалення з експлуатації агентів.

Модель керування зв'язками ПА (рис.2.15) складається з логічних компонентів, кожен з яких відображає набір властивостей ПА. Вони можуть комбінуватися у фізичній реалізації агентної платформи.

Агенти FIPA фізично існують на АР та використовують можливості, які пропонує АР для реалізації функціональності агентів. В цьому контексті агент як фізичний програмний процес має фізичний *життєвий цикл*, яким керує агентна платформа (рис.2.16). Для кожного агента цей фізичний життєвий цикл та пов'язані з ним стани можуть різнитися від зовнішнього логічного життєвого циклу та станів домену.

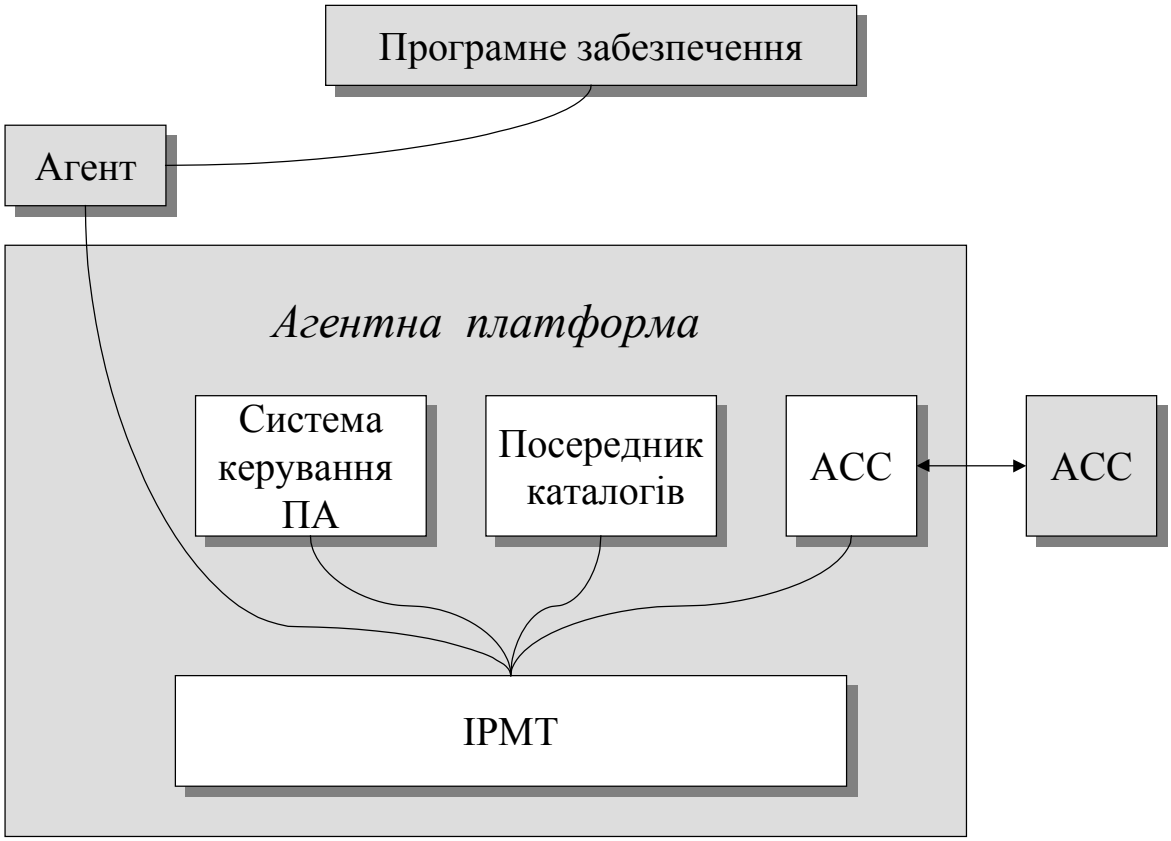


Рис.2.15. Модель керування зв’язками ПА

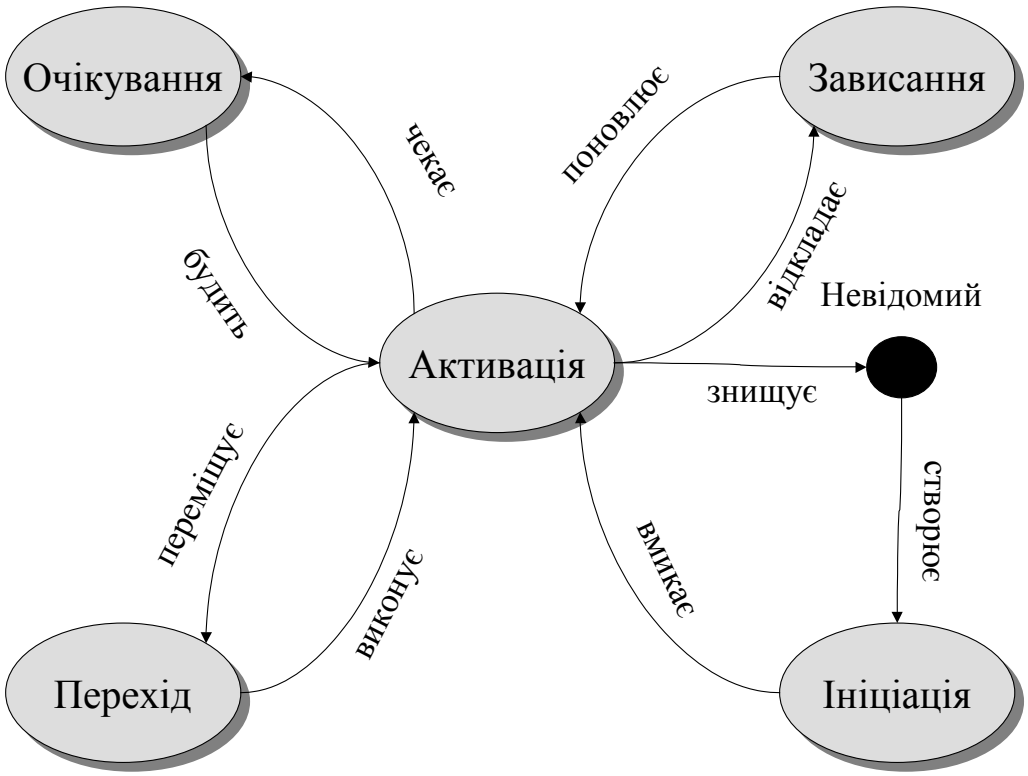


Рис.2.16. Життєвий цикл агента FIPA

FIPA визначає базовий протокол для інтероперабельності агентних платформ. Це означає, що завжди існує добре відомий метод для взаємодіє різних FIPA-сумісних агентних платформ (рис.2.17).

Життєвий цикл агента FIPA в АР:

- обмежується АР: агент фізично керується усередині агентної платформи, тому життєвий цикл статичного агента завжди обмежується специфікою АР;
- не залежить від застосунку: модель життєвого циклу не залежить від жодної прикладної системи та визначається лише станами та переходами сервісів агента в його життєвому циклі;
- орієнтований на екземпляри: агент, що описується в моделі життєвого циклу, розглядається як екземпляр (агент має унікальне ім'я та виконується незалежно);
- є унікальним: кожен агент має тільки один стан життєвого циклу АР в конкретний момент часу і тільки всередині однієї АР.

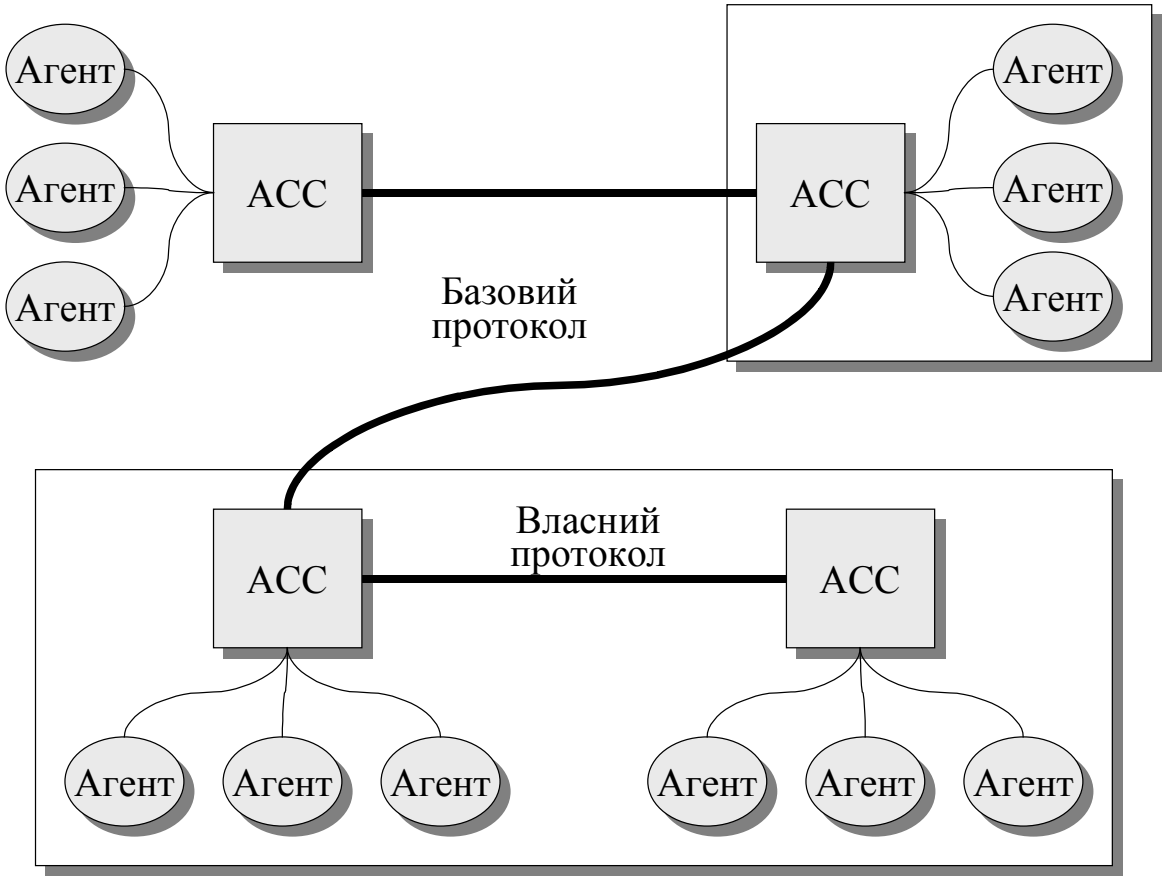


Рис.2.17. Базовий протокол FIPA

Внутрішній механізм транспорту повідомлень агентної платформи підтримує базовий протокол FIPA. Він має бути здатним розрізняти внутрішніх та зовнішніх одержувачів та знаходити найбільш придатний шлях доставки повідомлення.

АСС подає агента для керування комунікаціями на метарівні.

Однією з важливих характеристик ПА є здатність до гнучкого спілкування з іншими ПА. Цей напрямок інтероперабельності послужив основою проекту KSE (Knowledge Sharing Effort) для розвитку базової системи понять інтелектуальних систем.

Базовим елементом інтелектуальної взаємодії є пошук знань, що включає взаємне розуміння знань і спілкування (комунікацію) з приводу цього знання. Сутність є агентом тоді і тільки тоді, коли здатна до коректних взаємодій мовою комунікацій агента.

Мова комунікації агентів ACL – мова з точно визначеним синтаксисом, семантикою і прагматикою, що є основою комунікації між незалежно розробленими програмними агентами. *Комунікаційний акт* – особливий клас дій, що належать до базових будівельних блоків діалогу між агентами. Комунікаційні акти мають визначене, декларативне значення контексту. Вони моделюються на основі теорії мовних актів і подаються через посилання агента повідомлення іншому агенту з використанням формату повідомлень. В контексті комунікації *контент* – це частина комунікаційного акта, що представляє залежну від домена частину повідомлення. *Розмовою* називають послідовність комунікативних актів, коли агенти обмінюються повідомленнями за певною темою. При цьому може накопичуватися контекст (можливо, неявно), що використовується для визначення значення більш ранніх повідомлень. Одиницею комунікацій між агентами є повідомлення.

Теорія мовного акту (Speech Act Theory) – це теорія комунікацій, яка використовується як основа для ACL.

Щоб повідомлення одного ПА було зрозумілим іншим, вони мають приписувати однаковий зміст константам, що використовуються у цьому повідомленні.

Проблема спілкування більш високого рівня полягає у тому, що не існує загальновизнаної семантики для примітивів мови (актів мови, типів повідомлень). Стандартизована взаємодія і спілкування агентів буде нездійсненним доти, поки не буде існувати стандартизованого визначення значень, що мають специфічні повідомлення, якщо вони надіслані іншому агенту.

Мови комунікацій і їх властивості

Вимоги до мов комунікації агентів можна поділити на групи: до форми, змісту, виконання, роботи в мережі, середовища, надійності. Для рішення проблем спілкування між агентами у рамках проекту KSE з метою створення мови для підтримки різноманітних архітектур

агентів розроблено протокол KQML. При цьому особливий інтерес викликає спеціальний клас ПА – *посередники* (facilitators) комунікацій. Посередники – це агенти, що реалізують різноманітні комунікаційні сервіси, приміром, підтримують реєстрацію імен сервісів, переправляють повідомлення для них, виконують маршрутизацію повідомлень на підставі вмісту і забезпечують проміжний сервіс.

Подання знань ПА може бути поділено на дві підзадачі:

- переклад з однієї мови подання на іншу;
- пошук семантичного контексту подання знань для різних застосувань.

Комунікації викликають проблеми, які пов’язані з протоколом взаємодії, мовою комунікацій, транспортним протоколом.

Крім протоколів, ПА можуть містити інші компоненти, що допомагають виконувати свої задачі. Здатності міркувати про свої дії, представляти метазнання, планувати дії або моделювати інших агентів дозволяють підвищити можливості застосувань. Такі компоненти є периферійними для бази знань. Вони також можуть використовувати базу знань і комунікаційну мову.

KSE – це ініціативна група з розробки і розвитку технічної інфраструктури для підтримки пошуку знань між системами. KSE було організовано три робочі групи, кожна з яких вирішувала одну з проблем технології подання знань: Interlingua, Shared Reusable Knowledge Bases, External Interfaces. Група *Intrelingua* розробила загальну мову подання вмісту БЗ. Цією групою був опублікований документ, що описує формат обміну знаннями KIF (Knowledge Intrchange Format). KIF може бути використано для забезпечення перекладу з однієї мови на іншу або як загальну мову для двох агентів, що використовують різні мови подання (http://www.cs.umbc.edu/kse/kif/). Група *SRKB* (Shared Reusable Knowledge Bases) займалася розробкою розподілених БЗ і пошуком знань за визначеними темами, а також розробкою предметно-незалежних інструментів і методологій. Група *External Interfaces* розглядала питання взаємодії систем, заснованих на базах знань.

KIF є версією числення предикатів першого порядку з доданням підтримки немонотонних міркувань і визначень. Визначення мови включає опис синтаксису і семантики. KIF містить логічні оператори, які допомагають подавати логічну інформацію (заперечення, диз'юнкція, правила, квантифікаційні формули тощо). KIF дозволяє кодувати знання про знання, використовуючи символи "’" і ",",

оператори і пов'язаний покажчик. KIF можна також використовувати для опису процедур, тобто програм ПА. Семантика KIF (без правил і визначень) подібна логіці першого порядку. Вона розширюється за рахунок використання нестандартних операторів і обмежується аксіомами моделювання. За винятком цих розширень і обмежень, ядро мови зберігає фундаментальні характеристики логіки першого порядку.

KQML – мова і набір протоколів, що забезпечують ПА ідентифікацію, зв'язок і обмін інформацією з іншими агентами.

Основні риси KQML :

- повідомлення KQML непрозорі щодо вмісту;
- перформативи (performatives) – примітиви мови – визначають припустимі операції, які може здійснити ПА для комунікацій з іншими агентами;
- середовище агентів, що використовують KQML, може бути поповнене спеціальними агентами для виконання сервісних функцій: зв'язку фізичних адрес із символьними іменами, реєстрації БД, комунікаційних послуг тощо.

KQML складається з трьох рівнів:

- вмісту;
- комунікацій;
- повідомлень.

Рівень *вмісту* – це фактичний зміст повідомлення власною мовою подання (KQML може переносити висловлювання будь-якою мовою подання, у тому числі у вигляді рядків ASCII). Деякі ПА, що використовують KQML (приміром, програми маршрутизації, узагальнені посередники), можуть ігнорувати вміст повідомлення і тільки визначати, що воно закінчується.

Рівень *комунікацій* містить набір властивостей, що описують низькорівневі параметри комунікації (такі, як ідентифікація відправника й одержувача, ідентифікатор комунікації тощо).

Рівень *повідомлень* використовується для кодування того, що одна програма намагається передати іншій. Цей рівень формує ядро мови KQML і визначає вид взаємодії з агентом, що використовує KQML. Основною функцією рівня повідомлень є ідентифікація протоколу, що використовується для передачі повідомлення і забезпечення виконання певної дії, яку відправник зв'язує з повідомленням (запиту, команди тощо). Крім того, на цьому рівні можуть бути описані властивості мови вмісту, визначатися його тематика. Це дозволяє KQML виконувати аналіз , маршрутизацію і

доставку повідомлень навіть у тому випадку, якщо їхній вміст недоступний.

Синтаксис KQML ґрунтується на збалансованому списку з дужками. Початковим елементом списку є перформатив, а інші елементи є його аргументами у вигляді пар *(ключове слово, значення)*.

Агент KQML може вибирати для обробки кілька перформативів. Коаліція агентів може бути розширена шляхом використання додаткових перформативів, якщо вони погоджуються за своєю інтерпретацією і протоколами. Реалізація зарезервованих виконавців має здійснюватися стандартним способом (таб.2.6).

Таблиця 2.6

Категорії перформативів

Категорія	Ім'я
Основний запит (basic query)	Evaluate, ask-if, ask-about, ask-one, ask-all
Запит з багатьма відповідями (multi-response query)	Stream-about, stream-all, eos
Відповідь (response)	Reply, sorry
Загальна інформація (generic information)	Tell, achieve, cancel, untell, unachieve
Генератор (generator)	Standby, ready, next, rest, discard, generator
Визначення можливостей (capability-definition)	Advertise, subscribe, monitor, import, export
Робота в мережі (networking)	Register, unregister, forward, broadcast, route

Зарезервовані імена перформативів поділені на основні категорії. Крім перформативів стандартних дій комунікації (ask, tell, deny, delete) і протокольно-орієнтованих перформативів (subscribe), KQML має перформативи, які пов'язані з непротокольними аспектами роботи (приміром, *advertise* надає агенту повідомлення про те, які види KQML-повідомлень він може обробляти, *recruit* може використовуватися для пошуку ПА, здатних обробляти особливі типи повідомлень). Прикладом такої ситуації є запит клієнта до реляційної БД. Такий обмін вимагає, щоб сервер підтримував деякий внутрішній стан, індивідуальні транзакції, синхронні комунікації агентів.

Контекст комунікації – набір тверджень щодо оточення або знань ПА. Це відрізняє комунікації між ПА від взаємодій між іншими

програмними сутностями. Наприклад, стосовно пропозиції p можуть бути подані такі властивості, як "*переконаність у p*", "*переконаність у не p*", "*впевненість у p*" або "*впевненість у не p*".

Крім того, агенти здатні виконувати певні дії: здійснювати тільки свої наміри та впливати на інших агентів для виконання дій.

Вплив на дії інших агентів здійснюється спеціальним класом дій – комунікативними актами. Комунікативний акт здійснюється одним агентом стосовно іншого. Механізм виконання комунікативного акта такий же, як і в посилці повідомлення. Ролі ініціатора і реципієнта комунікативного акту позначаються як відправник і одержувач повідомлення. Разом з відповідним доменом знань комунікативний акт дозволяє одержувачу визначити контекст повідомлення.

Значення комунікативних актів подаються в термінах передумов, що належать до ментальних властивостей відправника, і очікуваних (з погляду відправника) наслідків ментальних властивостей одержувача. Однак, якщо відправник і одержувач незалежні, немає гарантії в одержанні очікуваних виведень.

2.5. Агентно-орієнтоване програмування

Термін "*агентно-орієнтоване програмування*" вперше запропонував Шохам для опису набору дій, необхідних для створення ПА. Агентно-орієнтоване програмування – різновид об'єктно-орієнтованого програмування з обмеженнями на параметри станів, типи повідомлень і методи. З урахуванням цієї точки зору агент – це об'єкт із певним "психічним" станом, який складається з ментальних компонентів. Шохам [208] формально описує стан агента у вигляді розширення стандартних епістемічних логік і визначає оператори рішень, здібностей і обов'язків. Програма агента керує його поведінкою і "психічним станом". Вона виконується інтерпретатором агентів. Комунікації між агентами здійснюються як примітиви різних типів.

Процес побудови ПА зводиться до наступних етапів:

- визначення функціональних задач, які має виконувати ПА;
- вибір мови програмування (приміром, .NET або Java) та інструментальних засобів реалізації ПА;
- створення концептуальної моделі ПА;
- створення логічної моделі ПА;
- створення фізичної моделі ПА та її програмна реалізація;
- тестування програмної реалізації ПА на відповідність заданим функціям.

Мови створення агентно-орієнтованих застосунків поділяються на кілька класів [162]:

- мови акторів – призначаються для створення колаборативних агентів (приклади – Actors);
- мови агентно-орієнтованого програмування – теж призначаються для створення колаборативних агентів (приклади – Agent-0, Placa);
- скриптові мови – призначаються для створення інтерфейсних, інформаційних та мобільних агентів (приклади – TCL/Tk, Safe-Tcl. Safe-Tk – http://www.smli.com/research/tct; Java – http://www.sun.com/; Telescript, Active Web Tools – http://www.genmagic.com/; Python – http://www.python.org; Obliq – http://www.research.digital/com/SRC/Obliq/; April; Scheme-48);
- реактивні мови – призначаються для створення реактивних агентів (приклад – RTA/ABLE).

Ключова ідея, що лежить в основі агентно-орієнтованої парадигми програмування, полягає у тому, щоб безпосередньо програмувати ПА через терміни ментальних понять.

Система АОП включає три компоненти:

- логічну систему для визначення ментального стану агентів;
- інтерпретовану мову програмування для створення агентів;
- процес "агентифікації" для поєднання програм агентів у низькорівневу реалізацію системи.

Інтерпретатор агентів забезпечує для кожного агента повторення через регулярний проміжок часу двох кроків:

- зчитування поточних повідомлень і відновлення "психічного стану";
- виконання дій, які приводять до зміни поведінки.

Система *Agent0* представляє собою першу спробу створення мови АОП. Логічний компонент цієї системи – мультимодальна логіка з квантифікаторами, яка дозволяє безпосередньо посилатися на час. Вона містить три модальності: переконання, виведення і можливість. Для подання ПА використовують:

- можливості (події, які агент може здійснювати);
- початкові припущення і висновки;
- правила виведення.

Основним компонентом, який визначає дії агента, є множина правил виведення. Кожне правило виведення містить умову повідомлення, ментальну умову і дію. Щоб визначити, чи потрібно виконувати певне правило, його умову повідомлення порівнюють з повідомленням, яке одержав агент. Ментальна умова порівнюється з

припущеннями агента. Якщо умови правила виконуються, тоді агент здійснює дію. Дії можуть бути персональними, які відповідають підпрограмам, або комунікативними, тобто мати посилання на повідомлення. Повідомлення належить до одного з трьох типів:

- запит на виконання дії;
- відмову у виконанні дії;
- повідомлення, що передає інформацію.

Результатом повідомлень запитів і відмов є модифікації дій агента, а результатом інформаційних повідомлень – зміни переконань агента.

Розроблений Шохамом інтерпретатор агентів Agent0 обробляє п'ять конструктів мови: фактичні висловлення, висловлення комунікаційних дій, висловлення умовних дій, змінні та правила обов'язків.

Agent0 є лише прототипом, що ілюструє принципи АОП. Базова концепція, сформульована Шохамом, вплинула в цілому на інші напрямки наукових досліджень агентів (PLACA, Concurrent METATEM, APRIL, MAIL, TELESCRIPT, ABLE тощо).

На сьогодні існує велика кількість ПЗ для проектування систем, що базуються на агентах. Необхідність у засобах програмної підтримки в цій галузі з'явилася ще в середині 1980- х років [71]. Поява великої кількості прототипів мов програмування агентів є ознакою того, що технологія агентів стала широко використовуватися.

Інструментальні засоби розробки агентів

Програмні агенти і МАС є потужною абстракцією для візуалізації та структурування складного середовища. Розробка ПА – складна та ресурсоємна задача.

На основі аналізу існуючих МАС можна сформулювати основні вимоги до інструментальних засобів розробки МАС [277]:

- забезпечення перенесення коду на різні платформи (UNIX, LINUX, Windows тощо);
- підтримка символьних обчислень;
- доступність інших платформ у гетерогенному мережному середовищі та підтримка мережної взаємодії між агентами, доступність віддалених ресурсів, інших мережних послуг (служби імен, RPC, OLE, CORBA, RMI тощо);
- багатопотокова обробка для реалізації одночасного виконання дій МАС, підтримка засобів синхронізації;

- засоби безпеки;
- об'єктна орієнтованість (механізми успадкування, виклики процедур як повідомлення, можливості синхронної й асинхронної взаємодії об'єктів, паралельність усередині об'єкта);
- підтримка специфічних для ПА властивостей, убудована в мову на рівні синтаксичних конструкцій (наприклад, відображення примітивами мови інтенсіональних характеристик об’єкта, інструкцій для переговорів тощо).

Для створення ПА, крім таких відомих понять, як процедури, функції, методи та класи, необхідно використовувати новітні технології розробки розподілених застосунків: програмування сокетів, виклики віддалених процедур RPC (Remote Procedure Call), об’єктні моделі розподілених компонент DCOM (Microsoft Distributed Component Object Model), віддалений виклик Java – Java RMI (Java Remote vocation) та архітектуру CORBA (Common Object Request Broker Architecture).

Java - об’єктно-орієнтована, розподілена, високопродуктивна, багатофункціональна, динамічна й універсальна мова програмування. У Java є убудована розширювана бібліотека класів, що містить модулі керування вікнами (AWT-Abstract Window Toolkit) для створення користувальницьких інтерфейсів, класи підтримки основних типів даних, багатопотоковості, мережних можливостей, графіки, мультимедіа тощо. Її базовий клас *CIAgent* широко використовується для створення платформно-незалежних ПА [22]. Найважливіші особливості Java:

- простота розробки застосування;
- переносність, що дозволяє Java-застосункам працювати на будь-якій апаратній платформі, яка підтримує віртуальну машину Java;
- масштабованість рішень.

Значна частина систем розробки МАС базуються на Java: JAT і JATLite, Afterburner, Infosphere, Java-To-go, Mole, Aglet, CyberAgent, Odyssey, Voyager, Concordia, MOA Java MAC тощо. Серед них тільки JATLite і Afterburner забезпечують обмін повідомленнями між ПА мовою KQML, що дозволяє їм через мовні актів виражати наміри розмови. Інші Java-орієнтовані системи спрямовані на мобільне керування та розподілені обчислення і не підтримують автономне спілкування між ПА.

Для створення інтелектуальних ПА найчастіше використовують такі мови:

- універсальні мови програмування (Java, C++);

- мови, орієнтовані на знання:
  - мови подання знань (KIF, OWL, DAML+OIL);
  - мови переговорів і обміну знаннями (KQML, AgentSpeak, April);
  - мови специфікацій агентів;
- спеціалізовані мови програмування агентів (TeleScript);
- мови сценаріїв і скриптів;
- символьні мови і мови логічного програмування.

На сьогодні створено велике різноманіття програмних засобів, призначених для розробки агентно-орієнтованого ПЗ, приклади яких – як комерційних (таб.2.7), так і вільно поширюваних (таб.2.8) – наведено нижче [4, 5].

Таблиця 2.7

Засоби розробки агентів

Назва продукту	Розробник	Мова	Опис
AgentBuilder® (www.agentbilder.com)	Reticular Systems, Inc.	Java	Інтегроване середовище для розроблення агентів та МАС
AgenTalk (www.kecl.ntt.co.jp/csl/msrg/topics/at)	NTT	LISP	Мультиагентні протоколи координації агентів
Agentx (www.ilk.com)	International Knowledge Systems	Java	Середовище розробки агентів
Aglets (www.trl.ibm.co.jp/aglets)	IBM Japan	Java	Мобільні агенти
Concordia (www.meitca.com/HSL/Projects/Concordia)	Mitsubishi Electric	Java	Мобільні агенти
DirectIA SDK (www.animats.com)	MASA Group	C++	Адаптивні агенти
Gossip (www.tryllian.com)	Tryllian	Java	Мобільні агенти
Grasshopper (www.ikv.de/products/grasshopper)	IKV++	Java	Мобільні агенти
iGENTM (www.cognitiveagent.com)	CHI Systems	C/C++	Інструментарій для розробки

Назва продукту	Розробник	Мова	Опис
			когнітивних агентів
Intelligent Agent Factory (www.bitpix.com)	Bits & Pixels	Java	Інструмент розробки ПА
Intelligent Agent Library (www.bitpix.com)	Bits & Pixels	Java	Бібліотека агентів
JACK Intelligent Agents (www.agent-software.com.au/ jack.html)	Agent Oriented Software Pty. Ltd.	JACK Agent Language	Середовище розробки ПА
JAM (members.home.net/marcush/ IRS)	Intelligent Reasoning Systems	Java	Агентна архітектура
Jumping Beans (www.jumpingbeans.com)	Ad Astra Engineering, Inc.	Java	Мобільні компоненти
LiveAgent (www.agentsoft.com)	Alcatel	Java	Побудова Інтернет-агентів
MadKit (www.madkit.org)	Madkit Development Group	Java, Scheme, Jess	Інструментарій розробки МАС
Microsoft Agent (msdn.microsoft.com/msagent/ default.asp)	Microsoft Corporation	Active X	Створення інтерфейсу
NetStepper (www.jwsg.com/netstepper)	JW's Software Gems		Середовище розробки ПА
Network Query Language (www.networkquerylanguage. com)	NQL Solutions		Мова програмування
Pathwalker (www.fujitsu.co.jp/hypertext/ flab/free/paw)	Fujitsu	Java	Бібліотека агентно-орієнтованого програмування
Swarm (www.swarm.org)	Swarm Development Group	Objective C, Java	Мультиагентне моделювання
Topia Personal Agents	Topia		Мобільні агенти

Назва продукту	Розробник	Мова	Опис
(www.topiaventures.com)	Ventures		
UMPRS (members.home.net/marcush/ IRS)	Intelligent Reasoning Systems	C++	Агентна архітектура
Via: Versatile Intelligent Agents (www.kinetoscope.com/via/ default.htm)	Kinetoscope	Java	Блоки побудови агентів

Таблиця 2.8.

Науково-дослідні проекти

Продукт	Дослідницька організація	Мова	Опис
Agent Building Shell (www.ie.utoronto.ca/EIL/ ABS-page/ABS-overview.html)	University of Toronto	COOL	Агентна архітектура
Agent Factory (krytan.ucd.ie)	University College Dublin, Ireland	Smalltalk-80	Середовище прототипів ПА
D'Agents (www.cs.dartmouth.edu/ ~agent)	Dartmouth University		Пошук, організація та презентація
Agent Tcl (agent.cs.dartmouth.edu/ general/agenttcl.html)	Dartmouth University	Tcl	Мобільні агенти
Architecture type-based Development Environment (samuel.cs.uni-potsdam.de/ soft/taxt/research/ade/ ade.html)	University of Potsdam	Java	Платформа та середовище розробки незалежних прикладних агентів
Ascape (www.brook.edu/es/ dynamics/models/ascape)	The Brookings Institution	Java	Структура ПЗ
Bee-gent (www2.toshiba.co.jp/ beegent/index.htm)	Toshiba Corporation Systems and Software Research Laboratories	Java	Структура розробки ПЗ

Продукт	Дослідницька організація	Мова	Опис
Bond Distributed Object System (bond.cs.purdue.edu)	Purdue University	Java	Структура агента
Cable (public.logica.com/~grace/ Architecture/Cable/public)	Logica Corporation	Agent Definition Language, C++	Архітектура системи
DECAF Agent Framework (www.eecis.udel.edu/ ~decaf)	University of Delaware	Java	Структура агента
DMARS (www.aaii.oz.au/proj/ dMARS-prod-brief.html)	Australian Artificial Intelligence Institute Ltd.	C, C++	Агентно-орієнтоване середовище розробки та реалізації
EXCALIBUR (www.ai-center.com/ projects/excalibur)	GMD First, Technical University of Berlin		Архітектура автономного ПА в комп-лексному середовищі комп'ютер-них ігор
GenA (www.crim.ca/~lmagnin/ pub/publis)	CRIM	GenA	Платформа мобільних агентів
Gypsy (www.infosys.tuwien.ac. at/Staff/lux/Gypsy)	Technical University of Vienna	Java	Мобільні агенти
Hive (hive.www.media.mit.edu/ projects/hive/hive.html)	The Media Lab Massachusetts Institute of Technology	Java	Інструмента-рій для побудови розподілених систем
InfoSleuth (www.mcc.com/projects/ infosleuth)	MCC	Java	Середовище колабора-тивних агентів
Infospiders	University of	Java	Адаптивні

Продукт	Дослідницька організація	Мова	Опис
(dollar.biz.uiowa.edu/ ~fil/IS)	California San Diego - Computer Science Dept.		розподілені інформаційні агенти
JADE (sharon.cselt.it/projects/ jade)	CSELT S.p.A., University of Parma	Java	Структура MAC
JAFMAS (www.ececs.uc.edu/ ~abaker/JAFMAS)	University of Cincinnati	Java	Структура MAC
JATLite (java.stanford.edu/java_agent/html)	Stanford University	Java	Пакети Java для MAC
JATLiteBean (kmi.open.ac.uk/people/ emanuela/JATLiteBean)	University of Otago	Java	Компонент JavaBean
JIAC (dai.cs.tu-berlin.de/english/forschung/projekte/JIAC)	Technische Universitat Berlin	Java	Агентна архітектура
Kasbah (ecommerce.media.mit.edu/ Kasbah/index.html)	Massachusetts Institute of Technology	unknown	Агент-посередник електронної комерції
KLAIM (music.dsi.unifi.it)	Universita' Di Firenze	Klaim	Ядро мови для взаємодії та мобільності агентів
Knowbot® System Software (www.cnri.reston. va.us/ home/koe)	CNRI	Python	Мобільні агенти
Mobiware Middleware Toolkit (comet.columbia.edu/mobiware)	Columbia University	Java	Мобільне мережне середовище
MOLE (mole.informatik.uni-stuttgart.de)	University of Stuttgart	Java	Мобільні агенти
Multi-Agent Modeling	Central European	MAML	Мова програ-

Продукт	Дослідницька організація	Мова	Опис
Language (MAML) (www.syslab.ceu.hu/maml)	University		мування
MultiAgent Systems Tool (www.gsi.dit.upm.es/~mast)	Technical University of Madrid	C++	Складні гетерогенні агенти
Open Agent ArchitectureTM (www.ai.sri.com/~oaa)	SRI International	C, C++, Prolog, Perl, Lisp, Java	Агентна структура
ProcessLink (cdr.stanford.edu/ProcessLink)	Stanford University	unknown	Агентно-орієнтована структура
RETSINA (www.cs.cmu.edu/~softagents)	Carnegie Mellon University		Агенти комунікацій
Social Interaction Framework (SIF) (www.dfki.de/~sif)	DFKI (German Research Institute for AI)	Java	Інструментарій МАС
Sodabot (www.ai.mit.edu/people/sodabot/sodabot.html)	MIT Artificial Intelligence Lab	unknown	Середовище розробки користувацьких ПА
SOMA (Secure and Open Mobile Agent) (lia.deis.unibo.it/Research/SOMA)	University of Bologna	Java	Середовище програмування ПА
TeamBots (www.teambots.org)	The Robotics Institute Carnegie Mellon University	Java	Мультиагентна мобільна робототехніка
TuCSoN (www.lia.deis.unibo.it/Research/TuCSoN)	Universita di Bologna		Модель координації Інтернет-агентів
Zeus (193.113.209.147/projects/agents/index.htm)	British Telecommunications Labs	Java	Середовище побудови агентів

Розглянемо детальніше деякі з цих програмних продуктів.

Комерційні продукти

AgentBuilder – інтегрований інструментарій, призначений для розробки інтелектуальних ПА та МАС. Він містить два основних компоненти: Toolkit та Run-Time System. Toolkit включає інструменти для керування процесом розробки агентно-орієнтованого ПЗ, аналізу домену, в якому має функціонувати ПА, розробки мережі для комунікацій між агентами, визначення поведінки окремих агентів та налагодження й тестування агентного ПЗ. Система Run-Time містить агентну машину, яка забезпечує середовище, в якому функціонують ПА.

Агенти, розроблені за допомогою AgentBuilder, взаємодіють один з одним за допомогою мови KQML. Крім того, AgentBuilder дозволяє розробникам визначати власні команди для комунікації. Всі компоненти AgentBuilder реалізовані на Java. Завдяки цьому розробка агентів може здійснюватися на будь-якій платформі та операційній системі, що підтримує Java. Агенти, створені в AgentBuilder, теж є Java-програмами та можуть виконуватися будь-якою віртуальною машиною Java.

Схема процесу розробки агентно-орієнтованих застосувань на базі AgentBuilder представлена на рис.2.12. Життєвий цикл ПА складається з таких етапів:

- обробка нових повідомлень;
- визначення, які правила поведінки можуть бути застосовані в поточній ситуації;
- виконання дій, що визначаються цими правилами;
- оновлення ментальної моделі відповідно до заданих правил;
- планування.

Власне ментальні моделі (початкова та поточна) включають опис вихідних (поточних) намірів, переконань, зобов'язань та можливостей специфікації правил поведінки.

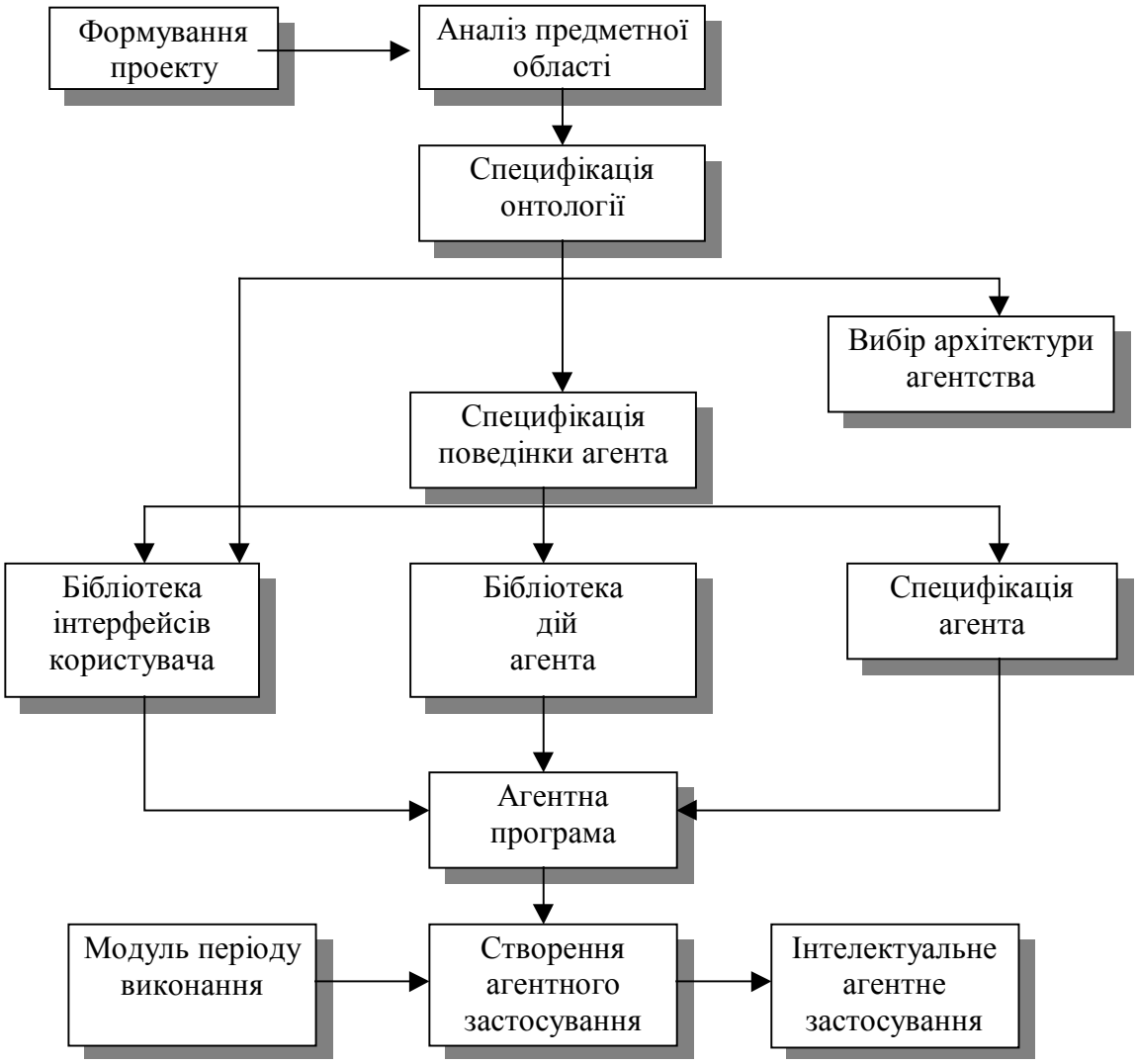


Рис.2.12. Етапи розробки ПА на базі AgentBuilder

Ця модель, що отримала назву RAMM (Reticular Agent Mental Model), є розвитком моделі Шохама [208], в якій всі дії виконуються тільки як результат певних зобов'язань. У RAMM ця ідея розширена до рівня загальних правил поведінки, що визначають дії агента в кожній точці його функціонування. При цьому правила поведінки фіксують множину можливих "відгуків" агента на поточний стан середовища так, як це передбачається цілепокладанням. Для специфікації поведінки агентів у системі AgentBuilder використовується спеціальна об'єктно-орієнтована мова RADL (Reticular Agent Definition Language). Правила поведінки можуть розглядатись як конструкції WHEN-IF-THEN. До МАС-бібліотек належать системи Bee-gent та JatLite.

AgenTalk – мова визначення координаційного протоколу МАС.

Agentx – набір бібліотек для розподілених обчислень у програмному середовищі Java. Ці бібліотеки компактніші, більш

функціональні та швидші за бібліотеки RMI, пов'язані з Sun JDK та реалізацією CORBA. Agentx не потребує використання мови визначення інтерфейсу IDL (Interface Definition Language). Він повністю сумісний з віртуальними машинами Sun, IBM та Microsoft, а також з останніми версіями браузерів Internet Explorer та Netscape Navigator. Крім того, Agentx забезпечує програмну підтримку для створення та реалізації автономних мобільних агентів. Об'єкти можуть переміщуватися мережею, під'єднуватися до хостів, розпочинати незалежне виконання свого програмного коду. Ця технологія дозволяє створювати досить складне ПЗ. Agentx може моделювати роботу великих мейнфреймів та суперкомп'ютерів, використовуючи робочі станції. Обчислювальні задачі, що вирішуються за допомогою Agentx, поділяються на локальні та розподілені у гетерогенній мережі задачі для одночасного, але незалежного виконання.

Аглет – об'єкт Java, здатний рухатися від одного хосту (будь-якого обчислювального пристрою, що підключений до мережі і використовує протоколи TCP/IP) до іншого. Якщо хост, на якому виконується аглет, припиняє його виконання, то він переключається на віддалений хост й продовжує виконання на ньому. Коли аглет переміщується, він переносить власний програмний код та дані, які він обробляє.

Технологія *Amzi!* розроблена на основі високорівневої мови програмування Prolog, призначеної для моделювання міркувань. Користувачам надається потужна бібліотека, яка легко переноситься та інтегрується з різними клієнтськими та серверними застосунками і середовищами.

Colony – система розподілених автономних ПА та застосунків, призначена для вирішення проблеми отримання організаціями найбільших переваг від використання гетерогенних динамічних IP, поданих у мережі – даних та знань. Враховуються різні формати подання даних, забезпечується мультиплатформність.

Concordia – структура для розробки та керування мобільними агентними застосунками, що забезпечують доступ до інформації на усіх пристроях, які підтримують Java. За допомогою Concordia застосунки можуть обробляти дані, доступ до яких надають локальні мережі та Інтернет, навіть тоді, коли користувач відключений від мережі.

DirectIA(r) дозволяє агентам або їх групам навчатися, прогнозувати та обирати дії. Такі агенти здатні адаптуватися до

динамічного середовища, знаходити нову поведінку та перевіряти свою ефективність. Машина застосунків DirectIA складається з бібліотек C++ для платформи Windows

Gossip – мобільний агент, що шукає інформацію в Інтернеті. Користувач повідомляє агента про свої інтереси, після чого агент шукає на сервері Marketplace агентів користувачів з тотожними інтересами та обмінюється з ними інформацією, яка зацікавила інших користувачів.

Grasshopper – перше середовище мобільних агентів, що відповідає стандарту MASIF (Mobile Agent System Interoperability Facility). Цей стандарт – надбудова над CORBA, що забезпечує інтеграцію клієнт-серверної моделі з технологією мобільних агентів. Використання цього стандарту забезпечує відкритість агентних застосунків для інших ПА у цьому середовищі.

Система розробки ПА *Intelligent Agent Factory* зменшує час, потрібний на створення інтелектуальних агентних застосунків. Такі агенти інтелектуальні у тому розумінні, що їх поведінка керується за допомогою правил на Jess, що використовується у CLIPS.

Intelligent Agent Library забезпечує структуру інтелектуальних агентів, яка містить засоби для комунікації агентів та побудови великих МАС. Ці агенти взаємодіють мовою KQML. Бібліотека також підтримує розробку мобільних агентів.

JACK Intelligent Agents забезпечує розробку архітектури ПА в розподілених застосуваннях. Система використовує JACK Agent Language (розширення Java).

JAM та *UMPRS* – інтенсіональні архітектури агента (Belief-Desire-Intention), що базуються на системі процедурних міркувань PRS. На відміну від більшості інших агентів, що застосовуються тільки в певних вузьких ПрО, JAM та UMPRS можуть використовуватися практично у всіх галузях. Ці архітектури підтримують міркування як зверху вниз, так і знизу вверх.

Madkit – вільно поширювана мультиагентна платформа, що базується на організаційних концептах. Вона використовує концептуальну модель AGR (Agent/Group/Role), в якій ПА розглядається як активна сутність, що взаємодіє через повідомлення з іншими агентами та відіграє певні ролі у групах. Така організаційна модель полегшує розробку МАС та дозволяє динамічно розробляти застосунки. Madkit, написаний мовою Java, працює в розподіленому середовищі на комп'ютерах з різними операційними системами та не потребує централізованого керування. Комунікації та групи ПА

можуть бути вільно розподілені. Для програмування ПА можуть використовуватися різні мови: Java, Scheme та Jess. Архітектура системи складається з модулів. Мікроядро, яке займає менше 60k коду Java, доповнюється різноманітними бібліотеками та ПА. Системні сервіси та інструменти налагодження теж реалізовані як агенти.

Microsoft Agent – набір програмованих сервісів, що підтримують подання інтерактивних елементів інтерфейсу Microsoft Windows.

NetStepper призначена для розробки та тестування ПА, які взаємодіють з Web-сторінками, серверами ftp та електронної пошти.

NQL – мова для створення інтелектуальних ПА та застосунків Web. Ця мова підтримує протоколи Інтернету та деякі інші сучасні стандарти.

Pathwalker – процесо- та агентно-орієнтована бібліотека Java. Вона підтримує корисні особливості розподіленого програмування: передачу асинхронних повідомлень з проміжним накопиченням, глобальні унікальні ідентифікатори процесів, засоби обробки мобільного коду та програмування дій агентів. Pathwalker працює з JDK1.1 та Java2 SDK (JDK1.2).

Swarm – ПЗ для мультиагентного моделювання складних систем. Система розроблялася як інструментарій для дослідників різноманітних дисциплін, пов’язаних зі штучним життям. Базова архітектура Swarm моделює набір одночасно взаємодіючих агентів. Swarm підтримує багато платформ: Solaris, GNU/Linux, Windows NT тощо.

Topia Personal Agents – клас ПЗ, що виконує персоніфіковані завдання. Система забезпечує запити на пошук, отримання, фільтрацію та доставку інформації, переговори щодо сервісів та продуктів, співробітництво з іншими агентами для виконання завдань та моніторингу діяльності агентів. Отримавши ціль, агент приймає рішення щодо здійснення завдань найкращим чином.

Система *Via* написана на Java та дозволяє розробникам вбудовувати властивості інтелектуальних ПА до існуючих застосунків та Web-сайтів. Via містить інструменти для створення задач агентів, логіки та користувацького інтерфейсу.

Voyager – агентифікований ORB (Object Request Broker). Він поєднує можливості мобільних автономних агентів та метод віддалених запитів з підтримкою CORBA. Voyager використовує синтаксис повідомлень Java для створення віддалених об’єктів, надсилання їм повідомлень та переміщення їх між застосунками. Він

дозволяє ПА самостійно переміщуватися та продовжувати виконання задач. Таким чином, ПА можуть діяти незалежно від імені клієнта, навіть якщо клієнт відключений від мережі.

Науково-дослідні проекти

Agent Building Shell забезпечує кілька повторно використовуваних шарів мов та сервісів для побудови МАС: координаційні та комунікаційні мови, керування основаними на дескриптивній логіці знаннями, розподіл корпоративної інформації, організаційне моделювання та вирішення конфліктів. Підхід використовується для розробки мультиагентних застосунків в галузі інтеграції ланцюга постачання виробничих підприємств.

Agent Factory – середовище прототипів агентів. Система написана мовою Smalltalk. Агенти – реактивні, соціальні, проактивні, автономні та інтенсіональні. Агентна модель підтримує конфігурацію ментальної архітектури. Для взаємодії агентів використовується мова Teanga ACL, але також підтримуються KQML та FIPA ACL .

Мета розробки системи мобільних агентів *D'Agents* – підтримка застосунків, що потребують пошуку, організації та подання розподіленої інформації.

Agent Tcl – інструментарій для розробки масштабованих МАС. Агенти створюються з використанням вбудованої мови Tcl (Tool Command Language). Мігруючі агенти шифруються та ідентифікуються за допомогою PGP (Pretty Good Privacy). Крім того, Agent Tcl підтримує обмін повідомленнями. Кожний ПА на конкретній машині має унікальні ідентифікатор і символьне ім’я. Для опису адресата вказують його машину та ідентифікатор. Агент Tcl має два компоненти – модифікований інтерпретатор, що виконує агентів Tcl, та сервер, що запускається на кожній машині, яка може отримати агента.

Об’єктно-орієнтоване моделювання ПА та МАС використовує найсучасніші досягнення у розробці ПЗ. Такий підхід описує програмну архітектуру як систему компонентів, взаємодія яких реалізується через з’єднання. Такі платформи, як IBM Aglets та Objectspace Voyager, не підтримують архітектурно-орієнтований підхід та середовище розробки агентів. Внаслідок цього такі платформи не забезпечують експліцитне моделювання взаємодії агентів або ж підтримують тільки обмежену кількість класів властивостей. Для вирішення цих проблем розроблена система *ADE* (Architecture type-based Development Environment).

AMETAS (Asynchronous MEssage Transfer Agent System) – система розробки мобільних ПА мовою Java. Основні переваги цієї системи – підтримка автономних ПА та повнофункціональна система безпеки. Для підтримки політики безпеки використовується повний спектр технічних особливостей Java, що робить AMETAS однією з безпечних систем створення ПА. Парадигма програмування агентів в AMETAS підтримує автономію через механізм комунікацій поштової скриньки. Система забезпечує робастість та масштабованість ПА. Крім того, додаткові властивості системи пов'язані з підвищенням інтерактивних можливостей агентів.

Ascape – ПЗ для розробки та аналізу агентно-орієнтованих моделей. Система забезпечує контекст для взаємодії агентів та набори правил, що визначають поведінку агентів.

ATOMIC (AgenT Ontology Manipulation Kernal) – компонент комунікації ПА, який підтримує мови ACL та CL, онтологічні подання та синтаксис. Продукт вільно розповсюджується.

Система *Bee-gent* – оболонка для розробки МАС, створена компанією TOSHIBA Corporation. У Bee-gent можна визначити поведінку обгортки ПА, яка надбудовується над застосунками за допомогою протоколу взаємодії. Bee-gent містить два типи агентів: обгортки Agent Wrappers використовуються для агентифікації існуючих застосунків, а посередники Mediation Agents підтримують координацію між застосунками через комунікації. Посередники переміщуються із одного сайту до іншого, де взаємодіють з агентами-обгортками. Поведінка системи базується на специфікації діалогу через протоколи взаємодії. Протоколи подаються як графи, основними поняттями яких є стани та переходи.

Протокол взаємодії визначає поведінку компонентів для досягнення цілі за допомогою взаємодії, що базується на діалозі (обміні повідомленнями) сукупності компонентів. Агенти-посередники виконують свої функції через взаємодію з обгорткою ПА шляхом переговорів. Обгортка ПА (агент-посередник) знаходиться у певному стані та виконує дії, умови яких відповідають поточному стану. Протокол взаємодії складається з понять "стан" та "перехід". Перехід – зміна одного стану на інший. Якщо агент виконує дію, то його стан змінюється за правилами переходу, що відповідає попередньому стану. Протокол взаємодії формується за допомогою визначення можливих станів, подій та правил переходу.

Базис повідомлень складає теорія мовних актів. Для подання повідомлень використовується спеціальна мова ACL/XML, яка розроблена на основі KQML.

Bond – розподілена об'єктна система, що забезпечує середовище обміну повідомленнями для розробки розподілених застосунків. Для взаємодії між об'єктами система використовує мову KQML. Агенти Bond можуть керуватися віддалено та взаємодіяти один з одним.

Cable застосовується для створення інтелектуальних ПА. Система надає користувачеві мову ADL для опису ПА та синтаксичний аналізатор Scribe для компіляції описів ПА з мови ADL в агентні застосунки.

Comet Way JAK надає просту модель програмування агентів, що спрощує розробку агентно-орієнтованих компонентів. Система забезпечує модульність, повторне використання та надійність компонентів.

DECAF (Distributed Environment Centered Agent Framework) забезпечує платформу для швидкої розробки ПА. Це здійснюється за рахунок надання середовища функціонування ПА, яке забезпечує інтерфейси, внутрішнє планування агентів та моніторинг їх дій як операційні примітиви. Базова архітектура DECAF використовує Java.

Агентна платформа *DIET* забезпечує створення МАС для розподілених обчислень. Низькорівневі розробки використовуються для масштабованості, робастості, адаптивності та розширюваності системи. Ця платформа призначається для швидкого створення прототипів застосунків.

dMARS™ – агентно-орієнтоване середовище для розробки складних розподілених систем. Це середовище забезпечує розробку, підтримку та реінженірінг системи. dMARS використовують для створення застосунків, що мають демонструвати проактивну цілеспрямовану поведінку та давати відповідь за фіксований проміжок часу. У системі застосовується високорівнева графічна мова планування для опису складних контекстно-залежних процесів. dMARS написана мовами C та C++.

EXCALIBUR – архітектура автономних агентів, які використовуються у складному середовищі комп'ютерних ігор. Такі агенти здатні знаходити вірні рішення для досягнення своїх цілей та адаптувати свою поведінку до змін середовища і дій опонентів. Крім того, ПА вміють взаємодіяти та координувати групові дії.

Агентна модель *GenA* дозволяє ПА функціонувати на різних платформах, використовуючи інтерфейс GenAi. Призначення моделі –

створення невеликих агентних застосунків. Програмування таких ПА не залежить від платформи.

GeneSyS (Generic Systems Supervision) призначається для створення інтелектуальних ПА з використанням Web-сервісів та SOAP. Це засіб моніторингу та керування розподіленими застосунками, що використовують Java, .NET та PHP. Для створення розподілених систем пропонуються стандартні рішення.

Gypsy використовує Java для створення гнучкого середовища програмування мобільних ПА. Такі ПА призначені для пошуку інформації в Інтернеті, електронної комерції, мобільних обчислень.

InfoSleuth реалізує співтовариство ПА, здатних до кооперативних дій. ПА досліджують та подають інформацію від особи користувача або застосунків, для чого пропонується простий інтерфейс. Інформація, що обробляється, може бути розподіленою та гетерогенною (приміром, поступати з кооперативних мереж або Web). Архітектура InfoSleuth складається з набору колаборативних ПА, які працюють разом за запитом користувача, який поступає з аплету. Поєднання запитів користувача утворює онтологію, яка характеризує ПрО його інтересів. Кожен запит обробляється відповідним ПА.

JADE (Java Agent DEvelopment Framework) – платформа для розробки агентно-орієнтованих застосунків відповідно до специфікацій FIPA щодо інтероперабельних інтелектуальних МАС. Мета розробки – спростити проектування МАС за допомогою набору системних сервісів та агентів. Можна розглядати JADE як посередника, що підтримує елементи функціонування ПА, які не залежать від конкретної області застосування: транспортні механізми, кодування, синтаксичний аналіз, життєвий цикл. Агентна платформа JADE 1.2 відповідає специфікації FIPA97 версії 1.2. та містить усі обов’язкові ПА. Агенти взаємодіють один з одним мовою FIPA ACL. Агентна платформа JADE надає базовий клас Agent, що реалізує функції взаємодії з агентною платформою (реєстрація, конфігурація, віддалене керування тощо), а також базовий набір методів для організації поведінки ПА (наприклад, відсилання повідомлень, використання стандартних протоколів взаємодії, реєстрація в кількох каталогах).

JAFMAS призначається для проектування МАС за допомогою класів агентів у Java. Система допомагає розробникам структурувати свої ідеї та реалізувати конкретний застосунок. Для взаємодії ПА використовуються KQML та інші подання мовних актів.

JASA призначається для проведення експериментів, пов'язаних з моделюванням аукціонів в агентно-орієнтованій інформаційній економіці. Програмне забезпечення надає базові класи для створення простих адаптивних торгівельних ПА.

Jason (*J*ava-based *a*gentSpeak interpreter used with *s*aci for multi-agent distribution *o*ver the *n*et) – інтерпретатор розширеної версії AgentSpeak, що підтримує комунікації між ПА, які базуються на мовних актах. Існують спеціальні реалізації системи для BDI. Ця система сумісна з іншими МАС з BDI-архітектурою.

JATLite є засобом створення МАС, що обмінюються повідомленнями. Цей набір пакетів Java спрощують створення МАС. JATLite складається з Java-класів та програм для створення систем ПА, які спілкуються через Інтернет для виконання розподілених обчислень. Система забезпечує базову інфраструктуру, в якій ПА реєструються в Agent Message Router, використовуючи ім'я та пароль, зв'язуються з Інтернетом, посилають та отримують повідомлення, передають файли тощо. Для взаємодії ПА використовується KQML, ACL та відкриті стандарти Інтернету: TCP/IP, SMTP і FTP. JATLite забезпечує шаблони комунікації, які користувач додає до ПЗ. Новизна наукового підходу, реалізована в JATLite, полягає у виконанні адміністративних функцій (реєстрації, з'єднання, ідентифікації агента) мовою агента та буферизації повідомлень для забезпечення поняття ідентичності агента. Інфраструктура JATLite подана на рис.2.13.

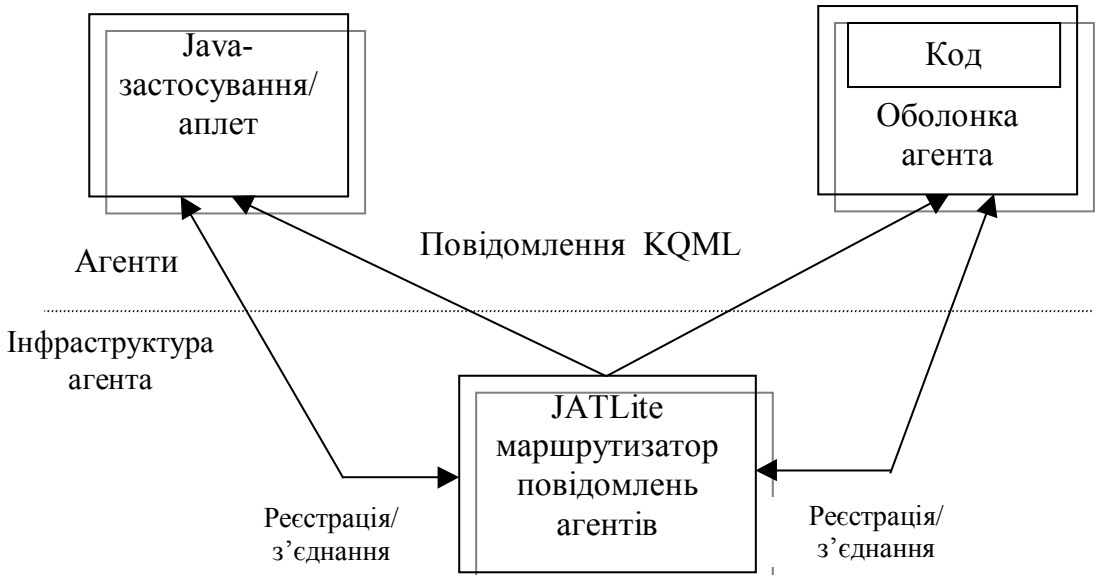


Рис. 2.13. Інфраструктура JATLite

Для розширення класів базових шаблонів JATLite використовує об’єктно-орієнтовані властивості Java, перебудовуючи їх для власних цілей. JATLite має модульну структуру (рис.2.14).

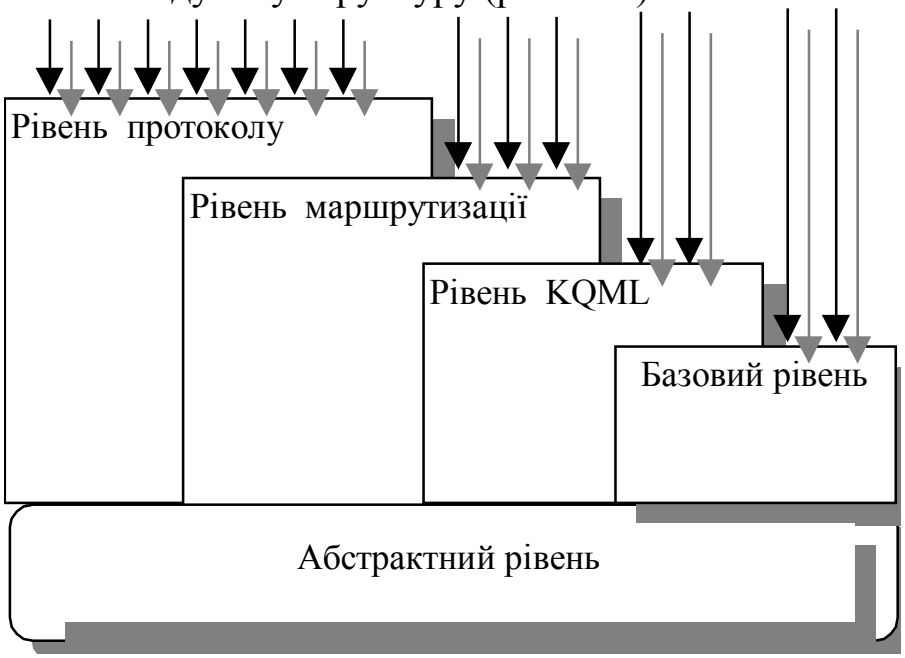


Рис.2.14. Архітектура JATLite

Абстрактний рівень забезпечує набір абстрактних класів, необхідних для застосування JATLite. Хоча використання JATLite передбачає, що усі з’єднання здійснені за допомогою TCP/IP, абстрактний рівень може бути розширено для інших протоколів, таких, як UDP. Базовий рівень забезпечує взаємодію на основі TCP/IP. Рівень KQML забезпечує збереження та обробку повідомлень. Рівень маршрутизації забезпечує реєстрацію імен, маршрутизацію повідомлень та обслуговування черги ПА через маршрутизатор повідомлень агентів. Рівень протоколу забезпечує кілька відкритих Інтернет-протоколів, таких, як FTP та SMTP.

Java Intelligent Agents Componentware – відкрита, масштабована й універсальна агентна архітектура. Її використання у Java забезпечує розробку компоненто-орієнтованих мобільних агентів, а також підтримує створення застосунків електронної комерції та розподілених телекомунікаційних сервісів.

Knowbot® – інфраструктура дослідження мобільних агентів ("Knowbot programs"), призначених для функціонування у розподіленому середовищі Інтернету. Для розробки Knowbot використана об’єктно-орієнтована мова програмування Python.

У проекті *Mole* також досліджуються мобільні агенти, надаються механізми їх міграції та комунікацій. Для створення ПА Mole використовує мову програмування Java .

Multi-Agent Modeling Language (MAML) – мова програмування для побудови агентно-орієнтованих моделей. Середовище MAML призначене для того, щоб допомогти дослідникам, які не спеціалізуються у програмуванні, використовувати комп'ютерне моделювання. Надаються засоби для того, щоб розробляти моделі, тестувати їх, здійснювати пошук у просторі параметрів та аналізувати результати моделювання.

MultiAgent Systems Tool забезпечує засоби кооперації гетерогенних ПА.

Open Agent Architecture™ (OAA) надає засоби інтеграції для співтовариства гетерогенних ПА, що діють у розподіленому середовищі. ПА визначаються як програмні процеси, що реєструють свої сервіси. Для взаємодії між агентами використовується мова ICL (Interagent Communication Language).

Метою проекту *ProcessLink* є створення засобів координації та взаємодії різнорідного ПЗ незалежних розробників. Основна ідея полягає у створенні оболонок для агентифікації ПЗ, які забезпечують обмін повідомленнями між застосунками.

Re:Agent – інтелектуальний ПА, призначений для обробки електронної пошти та здатний навчатися за власним досвідом. Використовує статистичні методи та алгоритми здобуття знань для автоматизації обробки контенту електронних листів.

RETSINA дозволяє розробляти структуру МАС. Система персоналізує обробку інформації, яку користувач отримує з Інтернету. Кожен агент у RETSINA складається з чотирьох модулів: комунікації, планування, складання розкладу та моніторингу виконання задач і запитів інших агентів. Усі модулі придатні для повторного використання.

SeMoA призначається для розробки мобільних агентів. Для підтримки функціонування ПА створено сервер. Цей сервер, а також ПА, розроблені за допомогою SeMoA, використовують Java (JDK 1.3). Система забезпечує захист агентів від зловмисних дій ПЗ хостів.

SeSAm (Shell for Simulated Agent Systems) забезпечує середовище для моделювання та експериментів з ПА. Використання SeSAm спрощує побудову складних моделей. Поведінка ПА визначається через UML-подібні діаграми. Користувач може явно задавати модель МАС. Використовуючи розширювану множину

примітивів будівельних блоків, користувач може візуально розробляти модель ПА замість того, щоб програмувати його традиційними мовами. Система має компонент адаптації параметрів.

Sim_Agent – інструментарій для розробки ПА зі складними гібридними архітектурами. Надаються механізми, які забезпечують керування розподілом ресурсів між компонентами ПА. Користувачі можуть застосовувати бібліотеки компонентів та готові архітектури, але жодні обмеження на архітектуру агентів не накладаються.

Sodabot – система для створення ПА. Її основні компоненти – схеми побудови агентів. У проекті пропонується оригінальна мова програмування для створення базових ПА, примітиви якої дозволяють користувачеві описувати діяльність агентів. Система орієнтована на створення персональних помічників та агентів розкладів зустрічей.

SOMA призначена для створення мобільних агентів, які відповідають вимогам масштабованості, динамічності, відкритості та безпечності, що є типовими для сценаріїв Інтернету. Основні цілі розробки – досягти безпечності та інтероперабельності. SOMA базується на фундаментальній моделі безпеки та надає широкий спектр інструментів і механізмів для побудови та реалізації гнучких політик безпеки. SOMA може взаємодіяти з різноманітними компонентами ПЗ, розробленими за допомогою різних стилів програмування. Система забезпечує інтероперабельність за рахунок сумісності з CORBA та MASIF. Крім того, SOMA надає абстракції розташування, необхідні для досягнення масштабованості, та забезпечує динамічну конфігурацію через Web-інтерфейси.

TeamBots – набір прикладних програм та пакетів Java для дослідження мобільних МАС. Середовище моделювання реалізовано мовою Java. Реалізація мобільних роботів іноді потребує низькорівневих бібліотек С, але для всіх високорівневих функцій використовується Java. TeamBots підтримує прототипування, моделювання та виконання систем керування роботами, які функціонують у середовищі TBHard, використовуючи програму моделювання TBSim.

TuCSoN – модель координації Інтернет-агентів. TuCSoN використовує центр кортежів – простір взаємодії, що базується на застосуванні кортежів. Цей центр реалізує координаційні правила, пов’язуючи поведінку агентів з комунікаційними подіями.

Zeus – середовище розробки колаборативних ПА. Кожен агент ZEUS складається з трьох шарів: визначення, організації та

координації. Шар визначення подає міркування ПА, його здатність до навчання, цілі, ресурси, здібності, переконання, переваги тощо. Організаційний шар описує відносини агента з іншими ПА. Шар координації описує технології координації та переговорів, які застосовує ПА. Комунікаційні протоколи побудовані над координаційним шаром та здійснюють комунікації між ПА. Під шаром визначення знаходиться шар інтерфейсу API.

Проблеми агентифікації програмного забезпечення

Агентифікація – це перетворення довільного ПЗ в ПА. Програмний агент може бути побудовано як оболонка поверх одного або кількох фрагментів програмного коду [212].

Кожен ПА забезпечує одну чи кілька корисних інформаційних послуг. Доступний і зрозумілий опис семантики цих послуг надається іншим агентам, що забезпечує ефективне функціонування мультиагентних систем.

Таку парадигму можна розглядати як розвиток *компонентного підходу* [280], який полягає в створенні програмних систем зі стандартизованих компонентів відповідно до визначених правил подання і маніпулювання програмними компонентами.

Програмний компонент (ПК) – довільний фрагмент програмного коду, що забезпечує виконання певних функцій. Повторно використовуваний компонент (ПВК) – це розширення ПК, що забезпечує його повторне використання в нових програмних розробках без залучення розробників [250].

Агентифікація ПВК переводить взаємодію з ним з рівня специфікації ПВК на рівень опису семантики послуг. Агент на відміну від людини-експерта має більш формалізований і докладний опис функціональності повторно використовуваних компонентів.

Проблеми цього процесу викликані гетерогенністю ПЗ: програми написані різними мовами, для різних платформ тощо. Системи розрізняються як синтаксично – через різні формалізми подання знань, так і семантично – однаково сформульовані знання можуть мати різне значення для різних систем.

Існує кілька загальних підходів до агентифікації раніше написаних програм:

- написання транслятора, тобто зовнішньої програми, що перехоплює повідомлення, які надходять і виходять із системи, і перекладає їх на комунікативну мову. Підхід використовується у

випадку, коли немає вихідних текстів системи, але є можливість перехоплення всіх повідомлень;

- створення оболонки системи, що входить до складу системи і виконує ті ж функції, що і транслятор. Такий підхід можливий, коли є доступ до вихідних текстів програм системи;
- повне переписування системи у випадку, коли інші підходи за якимись причинами неможливі.

2.6. Методології розробки мультиагентних систем

Агентна технологія останнім часом все ширше застосовується в ПЗ, і тому виникає необхідність у створенні методології для розробки мультиагентних систем.

Створення теоретичних моделей ПА та МАС потребує адекватної методології, яка дозволить справлятися зі складністю проблеми за допомогою її декомпозиції [171].

Метою етапу аналізу є перетворення вимог до системи в деяке абстрактне подання цієї системи. Аналіз системи має охоплювати те, *що* вона робить, а не те, *як* вона це робить. МАС через їх розподілену, кооперативну сутність мають характеристики, що вирізняють їх серед традиційних програмних систем, тому методи їх аналізу мають охоплювати такі властивості, як ролі агентів, протоколи їх взаємодії тощо. Метою етапу проектування є програмна реалізація системи, що відповідає раніше визначеним вимогам.

Зараз існує кілька підходів до розробки МАС, проте в них слабко висвітлено перехід від етапу аналізу до етапу проектування. Перехід від аналізу до проектування є критичним, тому що без чіткого визначення того, як це робиться, проектування буде не узгоджено з аналізом, і це може призвести до помилок у функціонуванні системи.

Методологія MaSE

Розвиток *MaSE* (Multiagent Systems Engineering) [133] – методології інженерії МАС – спрямований на те, щоб покрити закінчений повний життєвий цикл МАС. MaSE складається із семи кроків, показаних на рис.2.18. Перші три кроки відповідають етапу аналізу, а останні три – етапу проектування.

На етапі аналізу визначають цілі системи і ролі, що виконують ці цілі через набір взаємодіючих завдань. На етапі проектування визначають класи ПА, які виконують певні ролі, а переговори використовуються для опису деталізованих комунікаційних

протоколів. Проектувальник розробляє стратегію розгортання за допомогою діаграми розгортання, що уточнює, на яких платформах ПА постійно зберігаються і як здійснюються комунікації між ними.

Перший крок у MaSE – *визначення цілей* на системному рівні. Цілі визначаються як певні завдання і відображають те, чого система прагне досягти. Вони в загальному випадку залишаються постійними на всіх етапах процесу аналізу і проектування. Кожна дія в межах системи має підтримувати певну ціль. Після того як ролі ідентифіковані, кожній ролі призначають деякий набір цілей. На інтуїтивному рівні, якщо всі системні вимоги реалізовані як цілі, які виконуються ролями (які пізніше стануть агентами), система виконує необхідні початкові умови.

Наступний крок у MaSE – створення прецедентів, які описують послідовності подій та фіксують бажану поведінку системи. Кожний прецедент пов’язаний з конкретним випадком застосування системи. Слід використовувати як позитивні прецеденти, які описують те, що має відбуватися в процесі нормального функціонування системи, так і негативні, що описують бажану послідовність подій у випадку аварії або несправності.

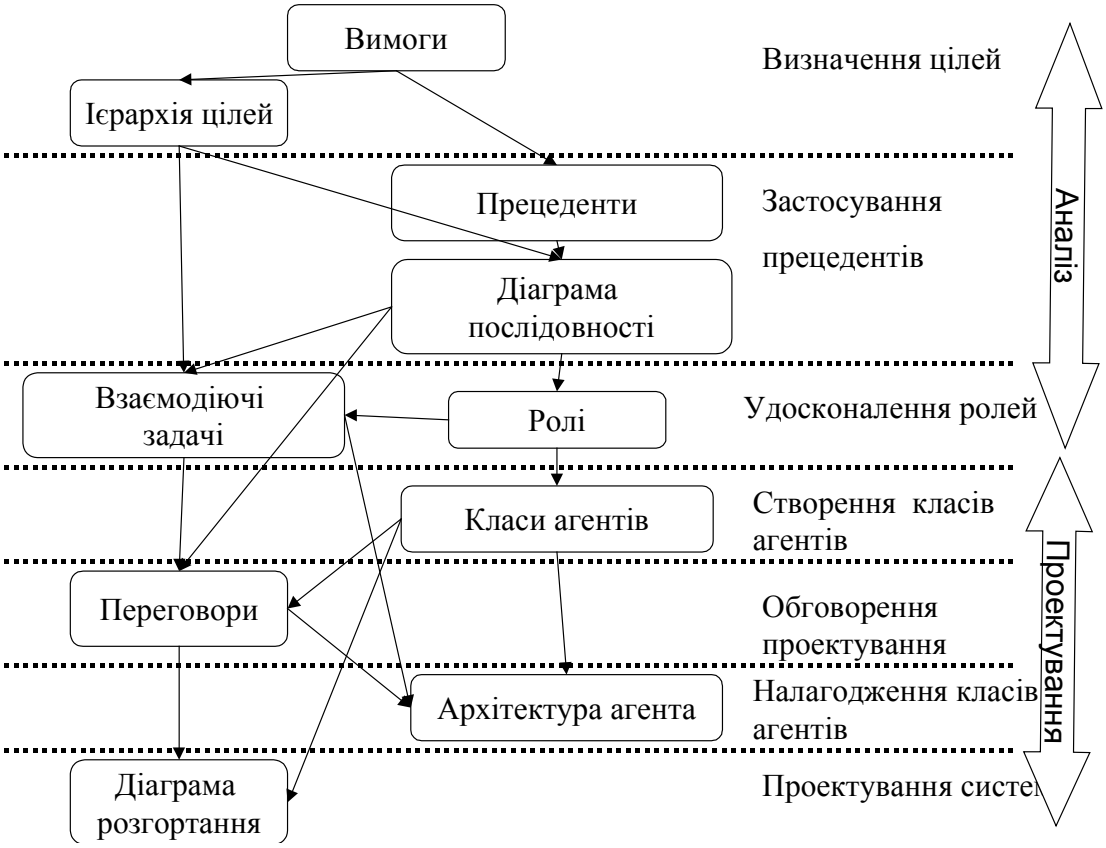


Рис.2.18. Етапи методології MaSE

Подальший крок – удосконалення ролей, де аналітик визначає, які ролі використовуються в системі і які задачі виконуються кожною роллю. За кожну ціль має відповідати принаймні одна роль. Ролі надають базу для створення класів агентів і подають системні цілі на етапі аналізу, сполучаючи те, що система робить, з тим, як саме вона виконує це.

Кожна роль може обслуговувати кілька спільно діючих задач, що визначають остаточну поведінку ролі. Взаємодіючі задачі використовуються для реалізації складних комунікаційних протоколів, таких, як Contract Net, Dutch Auction тощо. Це дозволяє користувачеві визначити, як елементи системи узгоджуються і взаємодіють один з одним. Ці задачі також лежать в основі переговорів між класами агентів на етапі проектування MaSE.

Задачі, що починають виконуватися одночасно після запуску ролі, продовжуються, доки роль не завершується або не досягнуто кінцевого стану. Важлива властивість задач – здатність взаємодіяти з іншими задачами для виконання власних цілей. Задачі можуть належати одній або різним ролям. Щоб задача могла зв'язуватися з задачею іншої ролі, потрібно за допомогою відправлення і одержання повідомлень визначити події, пов’язані із зовнішніми комунікаціями.

Створення класів агентів є першим кроком у фазі побудови MaSE. Класи агентів визначаються з ролей і на підставі цього створюється діаграма класів агентів (Agent Class Diagram), що описує класи агентів і взаємодію між ними на верхньому рівні. Клас агентів є шаблоном для агента, що функціонує усередині системи, і аналогом класу об'єкта в об’єктно-орієнтованому програмуванні.

Щоб переконатися у тому, що всі цілі системи досягнуті, кожна роль має бути виконана принаймні одним класом агентів.

Проектувальник має також визначити переговори, у яких беруть участь класи агентів. Якщо ролі А і Б визначені як паралельні задачі, що взаємодіють одна з одною, та агент X виконує роль А, а агент Y – роль В, тоді для забезпечення комунікації між агентами X і Y необхідно описати переговори для ролей А і В.

Результатом цього кроку є діаграма класу агентів, яка схожа на об’єктно-орієнтовані діаграми, проте агенти визначаються більше ролями, ніж атрибутами і методами, а всі зв'язки між класами є переговорами, що можуть мати місце між класами агентів.

Наступним кроком, визначеним у MaSE, є конструювання переговорів. Переговори є детальними погодженими протоколами між двома агентами. Вони складаються з діаграм класів комунікацій

ініціатора і респондента. Переговори – ядро будь-якої МАС, оскільки вони описують, як різні агенти спілкуються один з одним. Діаграми класів спілкування є скінченними автоматами, що визначають стани і переходи для кожної частини переговорів.

Ролі класів агентів визначають набір переговорів, у яких вони беруть участь. Деталі переговорів виходять із задач, пов'язаних з цими ролями. Переговори визначаються як прямий зв'язок між двома агентами, а кожна подія усередині діаграми класу комунікацій – як повідомлення іншого екземпляра агента, що бере участь у переговорах. Переговори не дозволяють одночасну комунікацію між кількома агентами або обмін між його внутрішніми подіями і внутрішніми компонентами.

Під час компонування класів агентів встановлюються внутрішні компоненти агента. Це процес складається з двох етапів: визначення архітектури агента і специфікації компонентів, які складають цю архітектуру.

Компоненти агента визначаються з використанням мови моделювання архітектури та об'єктної мови обмежень (Object Constraint Language). Це дозволяє користувачу визначити атрибути і функції агента. Кожному компоненту може бути поставлено у відповідність скінченний автомат, що визначає динамічні характеристики компоненти.

Останнім кроком, визначеним у MaSE, є розгортання системи. На цьому кроці проектувальник присвоює класам агентів значення, формуючи реальні агенти. Діаграма розгортання використовується для того, щоб показати детальну інформацію, необхідну для розгортання системи, включаючи значення, типи і локалізацію агентів усередині системи.

MaSE забезпечує керівництво для переходу з етапу аналізу на етап проектування в рамках повного процесу розробки. Моделі, що створюються на кожному кроці, чітко визначаються моделями, які створюються на попередніх кроках. У цій методології існують сильні зв'язки між моделями і є чіткі інструкції для здійснення переходу від аналізу до проектування, тому ця методологія досить перспективна для автоматизації.

Методологія Gaia

Методологія *Gaia*, запропонована Вулдріджем, Дженінгсом і Кіннеєм [91], – це сучасна розробка методології аналізу та проектування МАС, орієнтована на системи з відносно малою

кількістю (менше ніж 100) гетерогенних автономних агентів. Усі послуги, які надають агенти, і зв'язки, які вони мають з іншими ПА, розглядаються як статичні.

Верхній рівень абстракції відповідає етапу аналізу і складається з організації системи ролей, що мають зв'язки одна з одною і приймають участь у шаблонах взаємодій з іншими ролями.

Методологія Gaia розглядає систему як співтовариство, а елементи цього співтовариства – як ролі. Ролі спочатку описуються в моделі прототипних ролей, що поступово розширюється і детально розробляються до кінця фази аналізу.

Роль визначається за допомогою чотирьох атрибутів: *обов'язків (responsibilities), дозволів (permissions), дій (activities) і протоколів (protocols)*. Відповідальність поділяються на *властивості живучості (liveness properties*) і *властивості безпеки (safety properties*).

Обов’язки визначають, які функції мають реалізовувати ПА. Дозволи визначають ресурси, доступні ролям для досягнення їх обов'язків. У МАС такі дозволи звичайно є доступом до IP. Дії ролі – це обчислення, що можуть здійснюватися агентом без взаємодії з іншими агентами. Протоколи визначають порядок взаємодії ролей. Визначення протоколу складається з наступних атрибутів: *ціль (purpose), ініціатор (initiator), респондент (responder), входи (inputs), виходи (outputs*) і *обробка (processing*).

На етапі проектування потрібно створити моделі агента, його послуг і обізнаності. Методологія Gaia не містить методичного опису перетворення моделі проекту в проект, необхідний для реалізації системи. Без такої деталізації практично неможливо автоматизувати процес.

MAS-CommonKADS

Методологія MAS-CommonKADS [132] використовує об’єктно-орієнтовані підходи CommonKADS для МАС.

Аналітик визначає прецеденти, враховуючи початкові вимоги користувача. Метою цього кроку є фіксування ролей, після чого генерується специфікація системи за допомогою розробки моделей агентів, а також моделей їх задач, експертизи знань, організації, координації та комунікацій між ними.

Методологія описує, як ці моделі можна розвивати через керування ризиками. На першому кроці створюється початковий екземпляр моделі агента. На другому – задачі декомпозуються, для кожного агента визначаються цілі і підзадачі. Потім будується

координаційна модель, що описує взаємозв'язки і протоколи координації агентів. На четвертому кроці моделюються знання домена агентів, необхідні для виконання задач, а також способи їхнього поповнення і використання, переконання і припущення агентів про середовище. Останній крок полягає у побудові організаційної моделі.

Хоча в методології є деякі вказівки на те, як використовувати моделі аналітичного етапу для побудови моделей етапу проектування, але не визначено, як саме пов'язані ці моделі.

Методологія створення BDI-моделі

Створення моделей BDI-агентів, значно складніших порівняно з типовими об'єктами за внутрішньою структурою і за поведінкою, потребують розширення об'єктно-орієнтованих методологій, широко використовуваних для створення звичайних ІС.

BDI-агенти моделюються як складні об'єкти, які характеризуються своїми цілями, обов'язками, послугами, що вони забезпечують, інформацією, яку вони отримують і зберігають, і своїми зовнішніми взаємодіями. Для кожного агента треба змоделювати його інтенсіональні характеристики: переконання, бажання, наміри, цілі, плани тощо. Аналіз агентів базується на визначенні їх ключових ролей у своєму застосуванні, ідентифікація яких обумовлює ієрархію класів агентів. Визначення обов'язків кожного класу агентів дозволяє ідентифікувати послуги, які забезпечує та використовує агент, і його зовнішні взаємодії. Аналіз проблем створення і тривалості ролей і їх взаємодій визначає відношення керування між класами агента. Це фіксується в двох моделях: агента і взаємодії.

Модель ПА описує ієрархічні відношення серед різноманітних абстрактних і конкретних класів ПА, ідентифікує екземпляри ПА, що можуть існувати в межах системи, та задає обставини їх появи.

Модель взаємодії описує обов'язки класу агентів, послуги, які вони забезпечують, пов'язані взаємодії і відношення керування між класами агентів. Вона також фіксує синтаксис і семантику повідомлень, використовуваних для зв'язку як усередині агента, так і між агентами й іншими компонентами системи. Обидві моделі значною мірою незалежні від BDI-архітектури. Їх розробка складається з чотирьох основних кроків:

1. Ідентифікувати ролі агентів відповідної ПрО (наприклад, організаційні або функціональні) та тривалість життя кожної ролі.

Розробити ієрархію класів ПА. Початкове визначення класів ПА має бути високо абстрактним, не приймаючи будь-якого специфічного ступеня деталізації.

2. Для кожної ролі ідентифікувати пов'язані з нею обов'язки і послуги, які вона забезпечує або використовує для виконання своїх обов'язків. Послуги можуть містити взаємодію із зовнішнім навколишнім середовищем або користувачами. Наприклад, обов'язком ПА може бути контроль навколишнього середовища, відстеження певних подій і виконання дій, у тому числі повідомлення інших агентів або користувачів. І навпаки, обов'язок може вимагати, щоб агента сповіщали відносно умов або станів, виявлених іншими агентами або користувачами.
3. Для кожної послуги необхідно ідентифікувати взаємодії, пов'язані із забезпеченням послуг, перформативами і мовними актами, необхідними для цих взаємодій, та їх інформаційний зміст. Також треба ідентифікувати події й умови, які треба перевіряти, дії, що будуть виконуватися, й інші інформаційні вимоги, визначити відношення керування між агентами. На цьому кроці можна виконати внутрішнє моделювання кожного класу ПА, створивши моделі їх переконань, намірів та цілей.
4. Проаналізувати ієрархію агентів. Якщо є спільність інформації або послуг між класами агентів, тоді має бути створений новий клас агентів для спеціалізації існуючих класів. керуючись спільністю тривалості життєвого циклу, інформації, інтерфейсів та послуг і навести конкретні класи та екземпляри агента, потрібно створити класи агентів через спадкування або узагальнення.

Ролі, обов'язки і послуги призначені для опису цілеспрямованої поведінки на різних рівнях абстракції. Приміром, ролі можуть розглядатися як набори обов'язків, а обов'язки – як набори послуг. Послуги не підлягають подальшій декомпозиції.

Як тільки виконано декомпозицію ролей до рівня послуг і внутрішнього моделювання, модель агентства створено. Ролі служать відправною точкою для аналізу, а не визначенням того, що агенти отримають в результаті аналізу. Конкретні агенти можуть відображати групи послуг і обов'язків, що відрізняються від початкових ролей.

Така методологія є досить узагальненою, але вона задає порядок створення моделі МАС і може бути конкретизованою відповідно до призначення та сфери використання МАС.

2.7. Засоби інтелектуалізації поведінки програмних агентів

Для досягнення успіху в складному динамічному середовищі інтелектуальний ПА потребує засобів адекватного реагування на зміни в своєму оточенні або невідповідність апріорних припущень про це оточення. Поведінка агента описується *принципом раціональності* [157]: він обирає дію, якщо має знання про те, що ця дія приведе до однієї з його цілей. Агенти, що мають однакові знання і цілі, на рівні знань нерозрізнені, навіть якщо вони мають різну фізичну структуру.

Рівень інтелектуальності ПА визначається його здатністю до міркування і навчання, тобто можливостями здобуття необхідних знань за доступною інформацією. Критерії оцінювання самостійно здобутих знань збільшують автономність агента [32], а вміння ПА враховувати свій досвід спілкування з користувачами підвищує його ефективність. Тому засоби здобуття знань мають бути невід'ємною складовою моделей інтелектуальних ПА.

Система автономна (у розумінні "не знаходиться під безпосереднім контролем людини") у тій мірі, у який її поведінка визначається її власним досвідом.

Агент, що планує свої дії на основі вбудованих припущень, діє успішно тільки в тому випадку, якщо ці припущення виконуються, але цим припущенням може бракувати гнучкості. Внаслідок динамічності оточення ПА використання вбудованих припущень часто призводить до неефективних дій. Автономний ПА успішно діє навіть при великій розмаїтості оточення, маючи достатній час на адаптацію. Для цього він використовує елемент навчання, який за допомогою зворотного зв’язку порівнює переконання агента про доцільність виконання доступних дій з реальними результатами виконання цих дій. Якщо знаходяться розбіжності, то елемент навчання намагається внести зміни в переконання ПА. В процесі розробки елемента навчання треба враховувати, які переконання ПА потрібно удосконалити, як вони подані, як здійснюється зворотний зв'язок та яка апріорна інформація доступна ПА.

Можливість навчання є важливою властивістю будь-якої системи, яка функціонує у середовищі, що динамічно змінюється [29]. У МАС проблеми навчання особливо актуальні, але мають ряд особливостей. При цьому інтелектуальність розглядається вже не як властивість окремого ПА, а як властивість групи взаємодіючих агентів. Тому необхідно розглядати наступні питання:

- у чиїх інтересах має здійснюватися навчання (окремого ПА, групи агентів, всієї МАС);
- як здійснюється процес навчання (самостійно, через взаємодію з іншими ПА, усією групою);
- що є об’єктом навчання (поведінка ПА, взаємодія тощо);
- який метод навчання використовується;
- як перевіряти ефективність навчання.

Інтелектуальна поведінка системи пов'язана зі здатністю класифікувати об'єкти. Задача ПА полягає в тому, щоб за набором прикладів (з власного досвіду або досвіду інших ПА, з якими він здатний взаємодіяти) побудувати гіпотезу, яка узагальнює ці приклади або дозволяє знаходити рішення в аналогічних ситуаціях. Задача прогнозування властивостей полягає у визначенні невідомих характеристик об’єкта дослідження.

Використання механізмів навчання на базі прецедентів у МАС дозволяє забезпечити адаптивність їх поведінки. Механізми навчання можуть бути застосовані для вибору моделі поведінки агента, стратегії переговорів, протоколів взаємодії, прийнятті рішень тощо. Це пов’язано з використанням різноманітних методів ШІ: логіки, виведення на основі фактів, цілеспрямованого пошуку тощо. Розрізняють такі напрямки:

- розподілене виведення на основі фактів;
- колективне виведення на основі фактів;
- проактивне навчання;
- здобуття знань на основі аукціонів – координоване здобуття прецедентів у МАС.

Для прогнозування властивостей складних об’єктів використовують засоби індуктивного виведення (наприклад, за допомогою інструментального комплексу *ІндЕкс* [340]) та виведення за аналогією для структурно-атрибутивних моделей (приміром, таку задачу вирішує програмний комплекс "*Аналогія*" [274, 315]).

Якщо навчається один ПА, то він самостійно вирішує, чи є певний прецедент корисним для його навчання. У МАС приклад, який не потрібний одному ПА, може бути корисним для іншого. Тому такий процес навчання складається з двох етапів:

- акумуляції прецедентів;
- обміну прецедентами.

Для етапу акумуляції прецедентів виділяють три стратегії:

- агент ніколи на додає прецедент до своєї бази фактів;
- агент завжди додає прецедент до своєї бази фактів;

- агент додає прецедент до своєї бази фактів, якщо він є цікавим (приміром, якщо ПА невірно класифікує ситуацію, описану у прецеденті).

На етапі обміну прецедентами визначають стратегію поведінки ПА. Приміром, агент ніколи не пропонує прецедент для навчання іншим агентам або завжди пропонує його іншим ПА.

Агент A_i може отримати від агента A_j опис прецедентів, які можуть допомогти йому у вирішенні певної задачі, за наступним протоколом:

- A_i надсилає опис проблеми P іншим агентам $A_j, j=\overline{1,n}$, що входять до складу МАС;
- A_j здійснює спробу вирішити проблему, використовуючи свою базу прецедентів C_j;
- A_j надсилає A_i такі повідомлення:
  - відмову, якщо він не може вирішити проблему;
  - рішення у вигляді трійки $\langle S_k, P, A_j \rangle$, яке означає, що агент A_j вважає S_k найбільш ймовірним вирішенням проблеми P.

Схема голосування визначає механізм, за допомогою якого агент A_i будує узагальнене рішення на основі рішень, що надійшли від інших агентів $A_j, j=\overline{1,n}$. Приміром, узагальнене рішення може визначатися як клас, що отримав найбільшу кількість голосів (комітетна політика).

Висновки

Агентна парадигма привносить ряд принципово нових властивостей і можливостей в інформаційні технології і являє собою якісно новий рівень її розвитку, який забезпечується розподіленими обчисленнями в гетерогенному інформаційному середовищі Інтернету. Впровадження агентних технологій має підвищити ефективність інтелектуальної діяльності людини, позбавивши її від рутинних операцій. Одним з чинників інтересу до МАС став розвиток мережі Інтернету, в якій функціонують розподілені автономні програмні системи, що обробляють гетерогенні інформаційні ресурси.

Вибір агентних технологій як базових при проектуванні розподілених ІС забезпечує поєднання інтелектуальних засобів роботи з різними типами БД, БЗ і сховищ даних зі стандартизованими протоколами обміну знаннями. У такі системи ще на етапі

проектування закладаються гнучкість, горизонтальна і вертикальна масштабованість, відкритість архітектури та мобільність.

Делегування складних задач мультиагентним системам дозволяє представляти і вирішувати проблеми, які важко формалізуються. Можна виділити такі основні напрямки застосування агентних технологій:

- пошук, фільтрація та аналіз інформації;
- ефективний доступ до Web-сервісів;
- ситуаційне керування;
- дистанційне навчання;
- електронний бізнес;
- моніторинг даних;
- телемедицина;
- керування виробництвом у режимі on-line;
- керування транспортними потоками.

Застосування агентних технологій для пошуку інформації в Інтернеті розглянуто у наступній главі.

Глава 3. Застосування агентних технологій для пошуку інформації в Інтернеті

Людина освічена знає, де знайти те, чого вона не знає.
Г.Зіммель

У цій главі розглянуті засоби, які використовуються сьогодні для пошуку інформації в Інтернеті, їх переваги та недоліки. Проаналізовано типи інформаційних ресурсів, серед яких здійснюється пошук, та їх метаописи, засоби подання пошукових запитів та методи їх співставлення. Розглянуті переваги застосування агентних технологій для вирішення цієї проблеми.

3.1. Інформаційні ресурси Інтернету

Ефективність діяльності як окремих людей, так і організацій передусім залежить від отримання інформації, яку вони можуть використовувати в цьому процесі. На сьогодні одним з основних джерел інформації є глобальна мережа Інтернет – динамічне гетерогенне розподілене середовище.

Для упорядкування інформації, поданої в Інтернеті, постійно ведеться робота з виявлення закономірностей і розробки стандартів, які потім реалізуються в програмних продуктах. Існує тенденція перетворення Інтернету в мережу семантично пов'язаних IP, а потім і в розподілене сховище знань.

Типи інформаційних ресурсів Інтернету

Найпопулярніший сервіс Інтернет *WWW* (World Wide Web) – розподілена інформаційна мультимедійна система, що базується на гіпертекстовому поданні інформації. WWW забезпечує доступ до IP за допомогою трьох механізмів: єдиної системи іменування IP, протоколів доступу до IP та засобів гіпермедіа для ефективної навігації у цих ресурсах.

Будь-який IP, доступний в WWW, має унікальну адресу – *URL* (Uniform Resource Locator). Для доступу до IP використовують протокол *HTTP* (Hypertext Transfer Protocol) [65].

В Інтернеті подані різні види IP – текстові, графічні, аудіо та відео тощо (рис.3.1). Для публікації і поширення інформації в Інтернеті використовують мови HTML (Hypertext Markup Language [186]) та XML (Extensible Markup Language [26]), стандарти подання: графічної інформації JPEG, GIF; аудіо - MP3; мультимедійної – SMIL [216], Flash, MPEG [143].

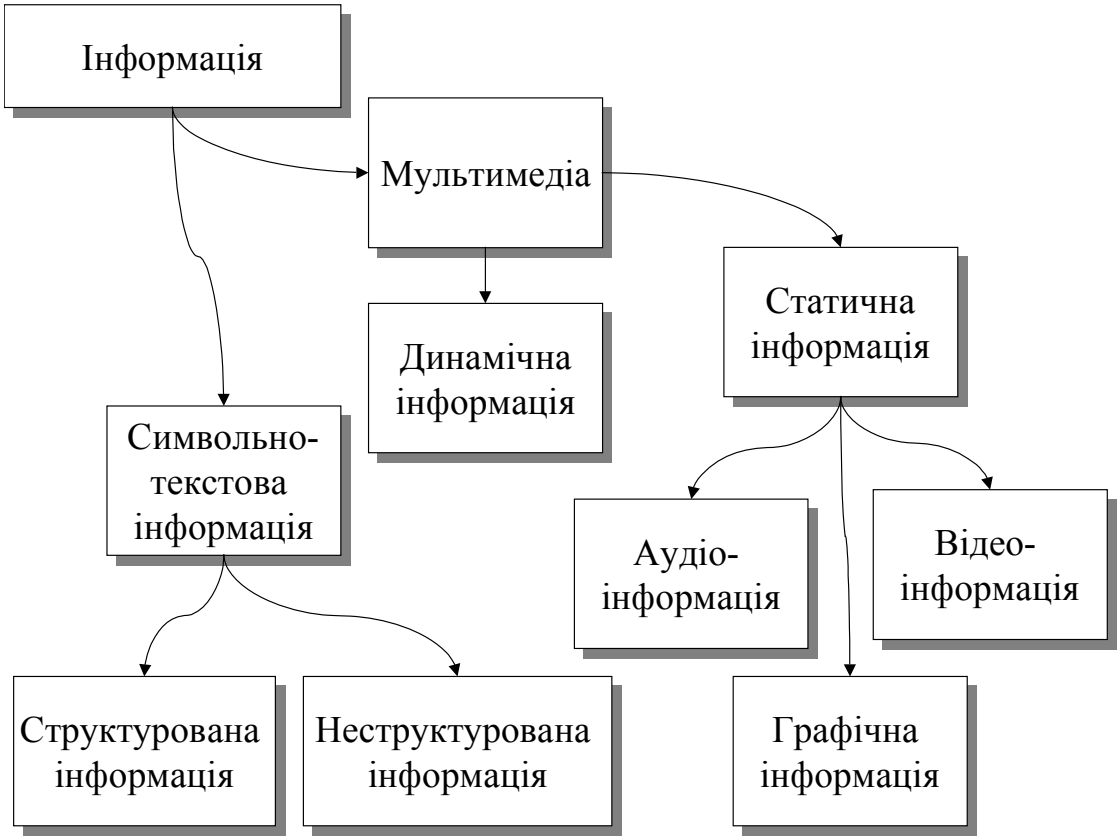


Рис.3.1. Класифікація IP Інтернету

Засоби подання текстової інформації

Поняття гіпертексту запропонував Буш ще в 1945 році. З 60-х років XX століття почали з'являтися перші застосунки, що використовували гіпертекстові дані. Але ця технологія набула поширення лише тоді, коли виникла реальна необхідність у механізмі об'єднання розподілених інформаційних ресурсів та забезпечення можливості обробки нелінійного тексту.

Інформація в WWW подається у вигляді документів, кожний з яких може містити гіперпосилання на інші документи, що зберігаються на різних серверах. За допомогою гіперпосилань користувач може швидко переміщатися між документами.

Найпопулярніша на сьогодні мова гіпертекстової розмітки HTML створена спеціально для подання інформації в Інтернеті. Це одна з ключових складових технології WWW. Використання гіпертекстової моделі документа впорядковує подання IP у мережі, а користувачі отримують зручний механізм пошуку і перегляду потрібної інформації.

HTML є спрощеною версією стандартної загальної мови розмітки SGML (Standart Generalised Markup Language) [211]. SGML –

засіб опису формальних специфікацій мов подання документів, структур документів і метаданих, затверджений ISO як стандарт ще в 80-х роках XX століття.

HTML припускає, що документ містить стандартні елементи розмітки, які відображаються стандартним чином: заголовок, параграфи, таблиці, цитування тощо.

Ускладнення структури документів та потреба у відображенні їх семантики призвели до того, що простота технології HTML з переваги перетворилася на недолік.

XML розроблений і прийнятий як стандарт у 1998 р. консорціумом W3C для забезпечення інтероперабельності між SGML і HTML. Це досить простий для розуміння й обробки підклас SGML.

Розширювана мова розмітки XML створена для текстового відображення структурованої інформації. Можна розглядати XML як метамову, яка не має власних операторів, але дозволяє описувати нові мови документів.

На відміну від HTML, що створювався для гіпертекстових документів з фіксованою структурою, XML призначений для розмітки документів довільної структури. У форматі XML можна зберігати дані практично будь-якої структури і призначення.

XML-документ за своєю структурою є деревом, яке можна інтерпретувати як граф. Він складається з елементів, а елемент може мати атрибути і значення, вкладений текст, дані і вкладені елементи. Перехід на специфікації XML дозволяє описувати структуру даних без прив'язки до форми їх відображення, визначати форму подання даних незалежно від конкретного змісту, створювати метадані та керувати доступом до даних тощо.

Розширивши множину тегів розмітки, розроблювачі XML досягли наступних цілей:

- явно виділили в документі структуру даних, що дозволяє подальшу автоматичну обробку документа, який при цьому усе ще залишається зрозумілим людині;
- відокремили дані, що містяться в документі, від того, як документ подається візуально, що розширює можливості для публікації документів на різних носіях.

Схема XML надає засоби для опису структури й обмежень, що накладаються на матеріали документів XML.

Імена в XML можуть належати різним просторам імен. Логічні схеми різних документів можуть використовувати ті самі імена елементів у різних значеннях. Для інтерпретації цих значень

необхідно вказати *простір імен* – колекцію імен, що ідентифікуються за посиланням URL, які використовуються документами XML як імена типів, елементів і атрибутів.

Простір імен можна розглядати як ресурс, з якого здобувають необхідні визначення [334]. Простори імен дозволяють розробникам XML-документів поєднувати кілька словників в одному документі для повного опису проблеми.

Крім гіпертексту, в Інтернеті досить часто застосовують інші формати для подання текстової інформації, приміром, у форматах PDF та DOC (технічні статті, науково-дослідні звіти).

Щоб здійснювати пошук серед таких документів, їх потрібно перетворювати у текстову форму (приміром, пошукова система Google може вести пошук у таких документах, перетворюючи їх в текстову форму для індексації).

Мультимедійна інформація

Значна частина IP, доступ до яких забезпечує сьогодні Інтернет, – мультимедійні . Параметри, які можна використовувати для опису мультимедіа, важко формалізовані та у значній мірі суб'єктивні. Проблеми подання й обробки мультимедіа привертають увагу багатьох дослідників, наприклад, ACM Multimedia Special Interest Group, Moving Picture Experts Group, Synchronized Multimedia Working Group консорціуму W3C.

Мультимедійний об'єкт відтворюється за допомогою обчислювальної техніки і впливає на одне або кілька людських почуттів: зір, слух тощо (слід зазначити, що основна частина інформації надходить до людини саме через зір і слух). В окремих випадках тактильна інформація також може відтворюватися за допомогою технічних пристроїв, наприклад тактильний монітор. У перспективі мають виникнути мультимедійні об'єкти, здатні впливати на інші почуття.

Обробка мультимедіа засобами обчислювальної техніки почалася ще в 60-х роках XX століття, коли зображення та текст були поєднані в одному документі. Термін "мультимедіа" має зараз багато визначень, але у більшості випадків це поєднання текстової, звукової та відеоінформації. Іноді під мультимедіа розуміють також потоки сигналів від спеціалізованого віддаленого обладнання (телескопів, датчиків тощо).

Під *мультимедіа* надалі розумітимемо інформацію, подану в електронній формі, яка не є символьно-текстовою та для інтерпретації

якої потрібне спеціалізоване ПЗ та відповідні периферійні пристрої виведення, що здатні впливати на різні органи почуттів користувача. Це визначення інтегрує кілька існуючих зараз концепцій специфікації мультимедійних даних.

З точки зору пошуку інформації пропонуємо таку класифікацію [279] мультимедійних IP (рис.3.2).

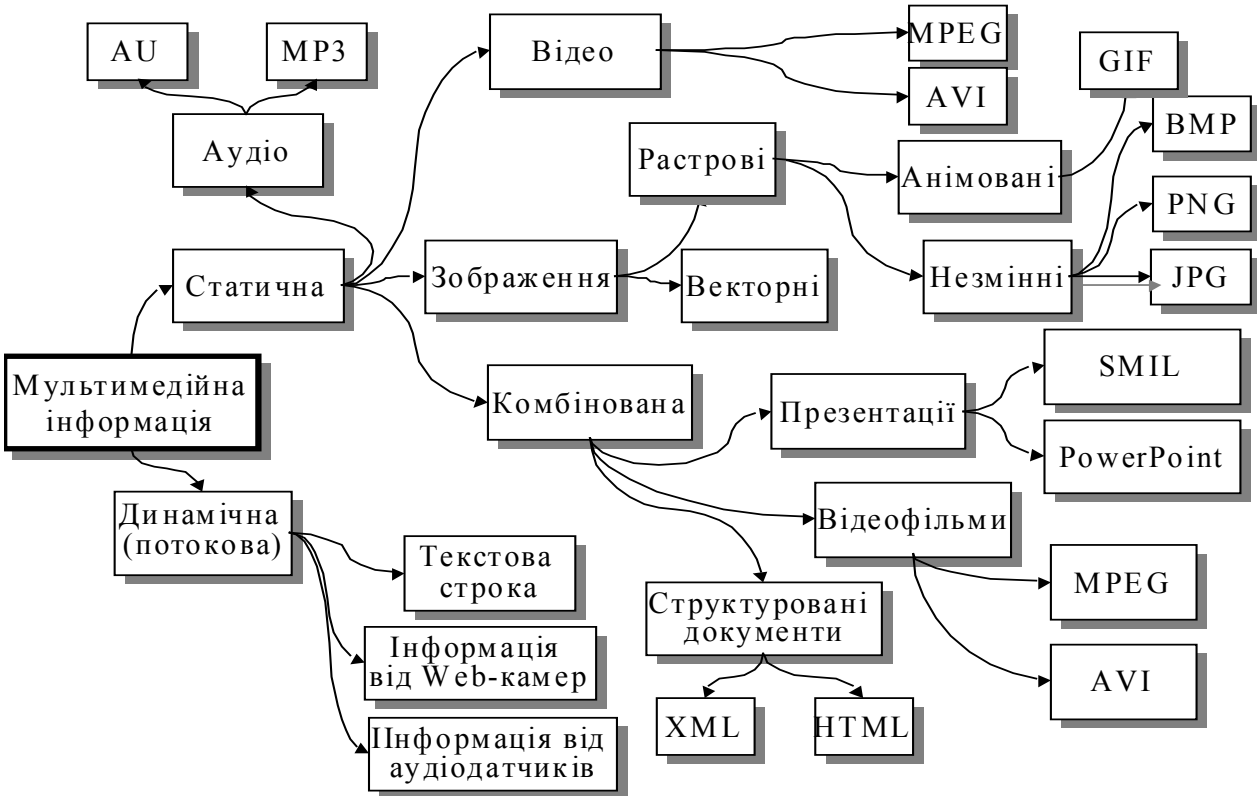


Рис.3.2. Класифікація мультимедіа

Зараз значні зусилля приділяються розробці універсальних засобів подання мультимедійної інформації. Експертна група Moving Picture Experts Group Об'єднаного Комітету зі Стандартизації запропонувала сімейство стандартів подання мультимедійної інформації MPEG [143].

MPEG-1 (ISO/IEC 11172) та *MPEG-2* (ISO/IEC 13818) [144] – стандарти стиснення зображення і звуку, призначені для запису інформації на CD-ROM і Video CD. Ці стандарти визначили розвиток відео-CD, MP3, DVD, цифрового телебачення тощо.

MPEG-4 (ISO/IEC 14496) [143] – стандарт стиснення рухомих зображень і звуку, що призначений для передання даних з низькою швидкістю, приміром телефонними лініями. Цей стандарт спрямований на об'єктно-орієнтоване подання мультимедійної інформації. MPEG-4 надає стандартизовані технологічні елементи для

інтеграції виробництва, розподілу і доступу до мультимедійного матеріалу.

MPEG-J – це розширення MPEG-4, в якому використовуються Java-елементи.

Стандарт *MHEG* (Multimedia & Hypermedia Expert Group) був прийнятий DAVIC (Digital Audio-Visual Council) для обміну мультимедійними об'єктами між застосунками та передачі їх різними способами (локальна мережа, мережі телекомунікацій тощо) з використанням MHEG object classes. Цей стандарт дозволяє програмним об'єктам містити в собі будь-яку систему кодування (приміром, MPEG), визначену в базовому застосуванні

MPEG-7 (Multimedia Content Description Interface – Інтерфейс для опису контенту мультимедіа ISO/IEC) [147] – стандарт, орієнтований на семантичне осмислення мультимедійної інформації.

Він базується на стандартах MPEG-1, MPEG-2 та MPEG-4 і використовує синтаксис XML Schema. Розробники MPEG-7 враховують такі стандарти, створені для досить вузьких доменів, як TV Anytime, Dublin Core [52], SMPTE Metadata Dictionary, EBU P/Meta та інші, і пропонують більш узагальнену модель. Зараз цей стандарт дороблюють у напрямку інтероперабельності з компонентами проекту Semantic Web.

Розробники MPEG-7 вважають аудіовізуальними даними статичні зображення, графіку, 3D моделі, аудіо, мову, відео, а також сценарії – інформацію про те, як ці елементи комбінуються в мультимедійному поданні. Засіб опису, запропонований у MPEG-7, не залежить від того, яка інформація описується (аналоговий відеозапис або оцифрований контент).

Основні засоби, що використовуються в описах MPEG-7, – мова DDL (Description Definition Language), схеми описів (DS) і дескриптори (D). DDL – мова схем для подання результатів моделювання аудіовізуальних даних. За основу DDL обрано XML.

Стандарт MPEG-7 складається з трьох частин:

- *засобів опису Description Tools;*
- *мови опису визначень DDL;*
- *системних засобів.*

Засоби опису Description Tools містять дві компоненти:

- *дескриптори* (Descriptors – D) визначають синтаксис і семантику кожної властивості (елемента метаданих);

- *схеми опису* (Description Schemes – DS) встановлюють структуру і семантику відношень між їх компонентами, які можуть бути як дескрипторами, так і схемами опису.

Вони поділяються на три категорії:

- *MPEG-7 Audio* – для аудіоматеріалу;
- *MPEG-7 Visual* – для відеоматеріалу;
- *MPEG-7 Multimedia Description Schemes* – для загальних характеристик мультимедіа.

DDL визначає синтаксис DS і дозволяє створювати нові схеми описів і дескриптори, розширювати існуючі схеми опису.

MPEG-7 у своїх описах використовує різний ступінь деталізації. Описи мають однозначно інтерпретуватися в контексті застосування, тобто той самий матеріал може бути описаний через різні типи властивостей, що відповідають сфері застосування.

Одне й те ж графічне зображення на найнижчому рівні абстракції може бути описане через форму, розмір, текстуру, кількість кольорів, палітру тощо.

Опис рис.3.3, це, приміром, – «*Чорно-біле зображення розміром 945 на 789 точок, яке займає у пам'яті обсяг близько 46 КВ і зберігається у файлі з ім'ям lad2.jpg*», тоді як на верхньому рівні буде подана семантична інформація, приміром «*Сцена з великою собакою, що ідентифікується як стафордширський тер'єр Келлі Лада де Мандрака, яка знаходиться праворуч, на фоні автомобіля, що не ідентифікується*».

Можуть існувати також проміжні рівні абстракції, зокрема, *"Чорно-біле зображення, на якому виокремлюються два об'єкти"*. Рівень абстракції пов'язаний зі способом здобуття інформації: багато низькорівневих властивостей можуть бути здобуті автоматично, тоді як високорівневі властивості зазвичай потребують втручання людини.

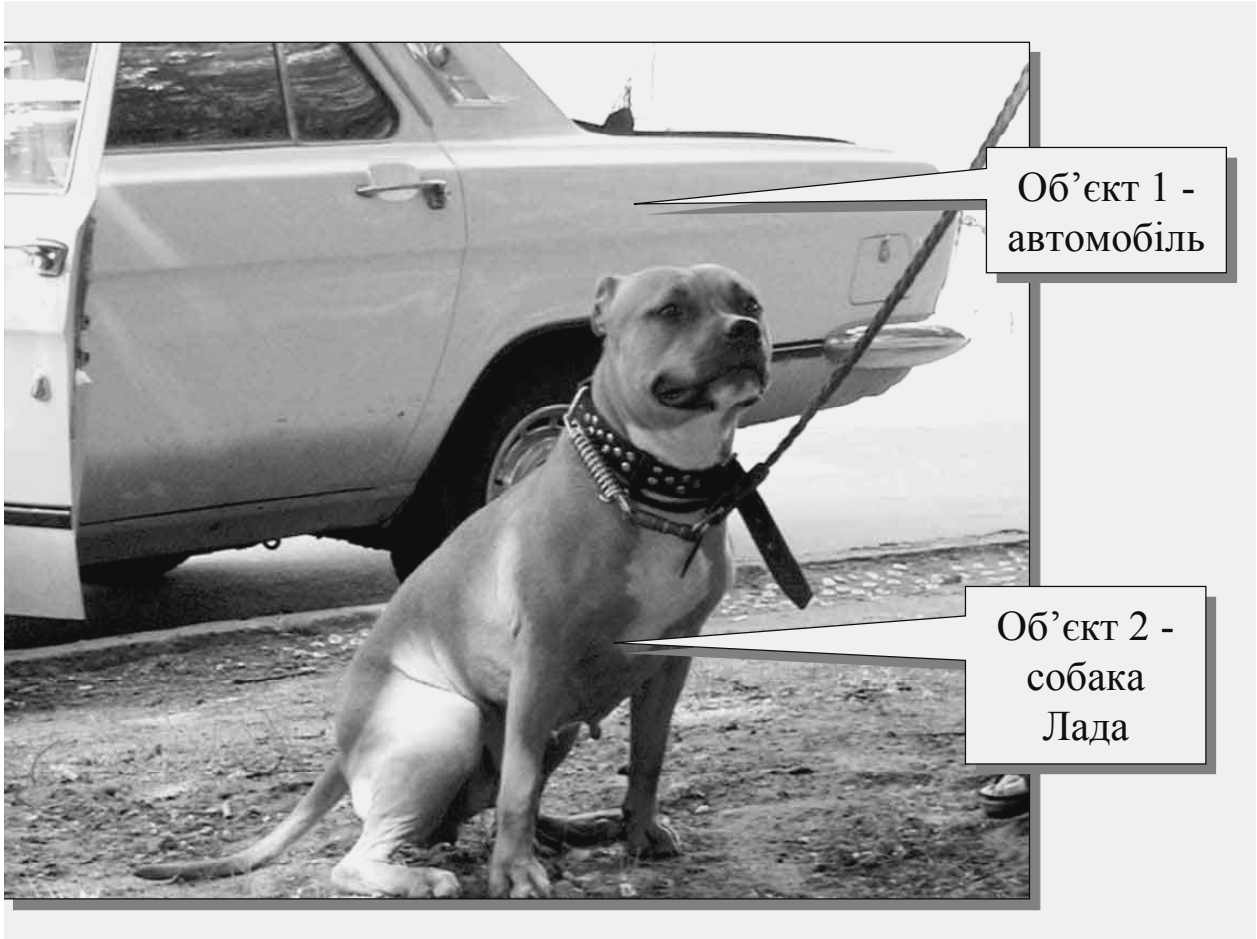


Рис.3.3. Зображення, яке може бути описане на різних типах абстракції

Для опису контенту мультимедійного IP використовують такі типи інформації:

- *форма* – формат кодування даних (JPG, MPEG-2 тощо), розмір даних;
- *умови доступу до матеріалу* – умови реєстрації, вартість доступу, угоди про права користування тощо;
- *класифікація* – оцінка походження матеріалу і тип його вмісту (обмежена кількість категорій задається заздалегідь);
- *посилання на інші релевантні матеріали*;
- *контекст* – обставини, за яких створено матеріал.

Засоби опису MPEG-7 Description Tools дозволяють створити описи контента (тобто набір схем опису DS та відповідних дескрипторів D), що містять інформацію про:

- створення і виробничі процеси;
- використання;
- особливості збереження;
- просторово-часові компоненти;
- низькорівневі властивості;

- дійсність, відображену в ньому;
- ефективні засоби перегляду;
- набір об'єктів;
- взаємодію з користувачем.

MPEG-21 [145, 146] – стандарт Multimedia Framework, призначений для створення інфраструктури керування контентом у розподіленому середовищі, яка забезпечує ефективний семантичний пошук інформації. Розробка стандарту спрямована на забезпечення інтероперабельності між усіма мультимедійними IP в Інтернеті.

Цей стандарт використовує такі терміни:

- *користувачі* (люди або програмні агенти);
- *цифрові елементи*, над якими користувачі виконують певні дії;
- *дії*, що генерують інші цифрові елементи, які можуть бути об'єктами транзакцій*;*
- *транзакції*.

MPEG-21 визначає основні елементи, необхідні для підтримки мультимедійних даних, залежності між цими елементами та операції, які вони підтримують, а також синтаксис та семантику елементів та їхніх характеристик, механізми їх збереження й узгодження тощо.

За стандартом MPEG-21, *користувач* – це певна сутність, що функціонує в середовищі MPEG-21 або використовує цифрові елементи (окремі особи, споживачі, організації, корпорації, консорціуми, уряди тощо). Користувачі ідентифікуються за їх відношенням до взаємодії з іншими користувачами, проте стандарт не розрізняє тих, хто надає контент, і споживачів інформації. Деякі користувачі можуть мати особливі права й обов'язки відносно взаємодії з іншими користувачами в межах MPEG-21.

Стандарт *MIME* (Multipurpose Internet Mail Extensions – багатоцільові поштові розширення Інтернету) [140] призначений для передачі інформації в мережі Інтернет з урахуванням типу цієї інформації. Він виділяє сім базових типів медіаінформації. Для кожного з них визначені параметри, характерні саме для цього типу інформації. У стандарті передбачена можливість розширення набору типів та їх параметрів.

Основні дискретні медіатипи MIME:

1. *Текст.* Текстова інформація.

2. *Зображення*. Зображення "Image" вимагає для перегляду інформації пристрій візуалізації. Основний підтип – розповсюджений формат подання зображень JPEG, що використовує кодування JFIF.

3. *Аудіо*. Контент підтипу "audio/basic" – одноканальне аудіо, закодоване з використанням 8bit ISDN mu-law [PCM] з частотою 8000 Hz.

4. *Відео*. Зображення, що змінюється у часі, і з узгодженим звуком. Основний підтип вказує на відео, закодоване відповідно до стандарту MPEG.

5. *Прикладні дані.* Бінарні дані, які обробляє певне ПЗ. Підтипи: "octet-stream" інтерпретується як бінарні дані, "PostScript" визначає PostScript-дані.

Вводяться також два композиційних базових медіатипи:

- *multipart* – дані, які складаються з кількох частин, що представляють собою незалежні типи даних. Основні підтипи:
  - "mixed", що визначає загальний комбінований набір частин;
  - "alternative" – використовується для даних у різних форматах;
  - "parallel" – для частин, які необхідно переглядати одночасно;
  - "digest" – для багаточасткових фрагментів, у яких кожна частина має тип "message/rfc822";
- *message* – інкапсульовані повідомлення.

Браузери Інтернету підтримують візуалізацію растрових зображень у форматах *JPG* (Joint Photographic Expert Group), *PNG* (Portable Network Graphics) та *GIF* (Graphics Interchange Format) [255]. JPG стискує повноколірні графічні дані з втратою інформації та звичайно використовується для подання фотографій, тоді як GIF стискує дані без втрати інформації. GIF може мати прозорі частини та припускає можливість створення анімації. Більш сучасна версія формату GIF – PNG.

Растровий формат, що використовують у сучасних цифрових перетворювачах зображень, передбачає кодування зображення у форматі по три байти на піксель, що призводить до створення громіздких, незручних в обробці файлів. Спеціально для цього формату розроблено багато схем, призначених для зменшення місця, яке файли займають на диску. Це особливо важливо, якщо графічні зображення мають передаватися по Інтернету.

Однією з таких схем є формат GIF, розроблений компанією CompuServe. Використаний у ньому метод полягає в зменшенні кількості відтінків кольорів пікселя до 256 і менше, у результаті чого колір кожного пікселя може бути поданий одним байтом замість трьох. За допомогою таблиці – палітри кольорів – кожний із припустимих відтінків пікселя асоціюється з певною комбінацією кольорів "червоний-зелений-синій". Змінюючи палітру, можна

змінювати кольори, що присутні в зображенні. Звичайно один із кольорів палітри GIF сприймається як прозорий.

Іншим прикладом системи стиснення зображень є формат JPEG. Це стандарт, розроблений асоціацією Joint Photographic Experts Group у рамках організації ISO. Формат JPEG показав свою ефективність для подання кольорових фотографій. Саме з цієї причини даний стандарт використовується виробниками сучасних цифрових фотоапаратів.

Стандарт JPEG містить кілька способів подання зображення, кожний з який має своє призначення. Наприклад, якщо потрібна максимальна точність подання зображення, використовується режим збереження інформації без втрат. У цьому випадку економія місця досягається за допомогою запам'ятовування розходжень між послідовними пікселями, а не яскравості кожного пікселя окремо. У більшості випадків ступінь розходження між сусідніми пікселями може бути закодована більш короткими бітовими комбінаціями, чим власне значення яскравості окремих пікселів. Існуючі розходження кодуються за допомогою коду перемінної довжини, що застосовується з метою додаткової економії пам'яті.

На жаль, при цього режиму створюються файли великого обсягу. тому він застосовується на практиці вкрай рідко. Більшість існуючих застосувань використовують інший стандартний метод формату JPEG – режим "базових рядків". У цьому режимі кожний з пікселів представляється трьома складовими: одним компонентом яскравості і двома компонентами кольору.

Зміст подібного поділу між кольором і яскравістю визначається тим, що людське око більш чутливе до змін яскравості, чим кольору. Режим "базових рядків" стандарту JPEG використовує цю особливість, кодуючи компонент яскравості кожного пікселя, але усереднюючи значення колірних компонентів для блоків, що складаються з чотирьох пікселів, і записуючи колірні компоненти тільки для цих блоків.

У результаті подання зображення зберігає різкі перепади яскравості, однак залишає розмитими різкі зміни кольору. Перевага цієї схеми – в тому, що кожен блок з чотирьох пікселів подається тільки шістьма значеннями, а не дванадцятьма, які у схемах з трьома показниками для кожного пікселя.

Додаткової економії місця можна досягти за допомогою запису інформації, що визначає зміни компонентів яскравості і кольору, а не їхніх абсолютних значень. У цьому випадку, як і в режимі "без втрат"

формату JPG, відправною точкою є той факт, що при скануванні зображення рівень розходжень між сусідніми пікселями може бути закодований з використанням меншої кількості бітів, чим при записі самих характеристик окремих пікселів (ці зміни кодуються за допомогою математичного методу дискретного косінусного перетворенням, який застосовують до блоків розміром 8х8 пікселів). Така бітова комбінація додатково стискується з використанням кодів перемінної довжини.

Режим "базових рядків" формату JPEG дозволяє отримувати кольорові зображення прийнятної якості, розмір яких знаходиться в співвідношенні 1:20 з розміром растрових файлів, у яких для подання кожного пікселя використовується трибайтова схема. Режим "базових рядків" формату JPEG дозволяє істотно скоротити розміри файлів за рахунок незначного і практично непомітного зниження якості зображення. Це робить цей формат популярним серед користувачів. Однак у деяких випадках використання інших методів дає кращі результати. Приміром, формат GIF дозволяє краще подавати зображення, що складаються з блоків одного кольору з чіткими границями (як у кольоровій мультиплікації).

Широко розповсюдженим способом *кодування аудіоінформації* [255] (з метою її збереження та обробки) є вимірювання значення амплітуди звукової хвилі через регулярні інтервали часу і запис послідовності отриманих значень. Щоб отримати необхідну якість звучання, на сучасних компакт-дисках музика записується з частотою вибірки 44 100 значень на секунду. Кожне зі значень записується в 16-розрядному форматі (для стереозапису використовується 32-розрядний формат). Отже, для збереження звукових даних із тривалістю звучання в одну секунду потрібно більш мільйона бітів пам'яті. Подібні витрати пам'яті прийнятні для запису музики на компакт-дисках, однак у сполученні з відеозаписом (для отримання озвучених зображень, що рухаються) ці вимоги перевищують можливості сучасних інформаційних технологій. Тому асоціація MPEG, що входить до складу ISO, розробила методи стиснення аудіоінформації, які дозволяють істотно знизити вимоги до використання пам'яті. Одним з таких форматів є MP3 (MPEG-1, Audio Layer-3), що дозволяє стискати аудіоінформацію у співвідношенні 12:1. При використанні цього формату музичні записи мають такий обсяг, який дозволяють ефективно пересилати їх мережею Інтернету.

Розглядаючи засоби подання мультимедійних ресурсів Інтернету, слід також відмітити розробки в об’єктно-орієнтованого

підході до подання мультимедіа, зокрема, такі графічні формати, як SMIL (Synchronized Multimedia Integration Language) [216], SVG (Scalable Vector Graphics) [205], VML (Vector Markup Language) [227], VRML (Virtual Realty Modelling Language) [232], PGML (Precision Graphics Markup Language). Більшість з них використовують подання інформації засобами XML. На відміну від растрової векторна графіка зберігає зображення як набір об’єктів (ліній, фігур та ін.), що дозволяє змінювати розміри зображень без втрати його якості.

Специфікація *SVG* 1.0 [2] виникла як результат розвитку таких стандартів, як VML (Vector Markup Language) компанії Microsoft і PGML (Precision Graphics Markup Language) компанії Adobe. Цю специфікацію підтримують такі розробники ПЗ, як Adobe, Apple, Autodesk, BitFlash, Corel, HP, IBM, ILOG, INSO, Macromedia, Microsoft, Netscape, OASIS, Open Text, Quark, RAL (CCLRC), Sun, Visio, Xerox, а також консорціум W3C. SVG базується на мові XML. Форма і рух об'єктів SVG задаються тегами і керуються за допомогою атрибутів, а для їхнього подання на екрані використовуються стилі. Завдяки цьому вдається зменшити розміри як простих, так і складних зображень.

Мова *VRML* призначена для опису тривимірних зображень і оперує об'єктами, що характеризують геометричні фігури та їхнє розташування в просторі. VRML-об’єкти називають вузлами. Їх використовують для опису побудови тривимірних об’єктів та правил їх руху. VRML-документ – текстовий файл, який інтерпретується модулем, що розширює можливості звичайного браузеру.

SMIL – рекомендований консорціумом W3C механізм створення документів, що містять синхронізовану мультимедійну інформацію [216]. Такі документи називаються SMIL-презентаціями. Вони включають набір інструкцій, які описують текстові дані, відео і аудіо та визначають послідовність їхнього відтворення. За допомогою SMIL можна стандартним способом поєднувати і синхронізувати різні елементи, такі, як текст, графіка, звук і відео. Основні сфери застосування SMIL – мережне телебачення, відео та дистанційна освіта. Фрагменти SMIL-презентацій можуть включатися у звичайні HTML-сторінки і відтворюватися або спеціально призначеними для цього програмами перегляду, або модулями розширення браузерів. SMIL надає наступні можливості:

- синхронізація примітивів: документи на SMIL описуються з використанням тегів паралельного та послідовного відтворення;
- необмежене повторне використання мультимедійних об'єктів;

- можливість використання аудіозаписів різними мовами;
- вибір смуги пропущення.

Розглянувши поширені засоби подання об'єктно-орієнтованої графічної інформації, можна зробити такі висновки: усі ці специфікації мають приблизно однакові можливості, використовують синтаксис мови XML для опису графічних векторних об'єктів. Тому у майбутньому можна прогнозувати їх поєднання або створення універсального браузера для перегляду інформації у різних форматах. Основним недоліком усіх цих специфікацій є відсутність зручних засобів створення ІР.

Структуровані джерела інформації

Всі інформаційні ресурси, доступні через Інтернет, можна класифікувати не тільки за типом даних (текст, зображення, звукові дані, відео тощо), але й за способом зберігання (рис.3.4).

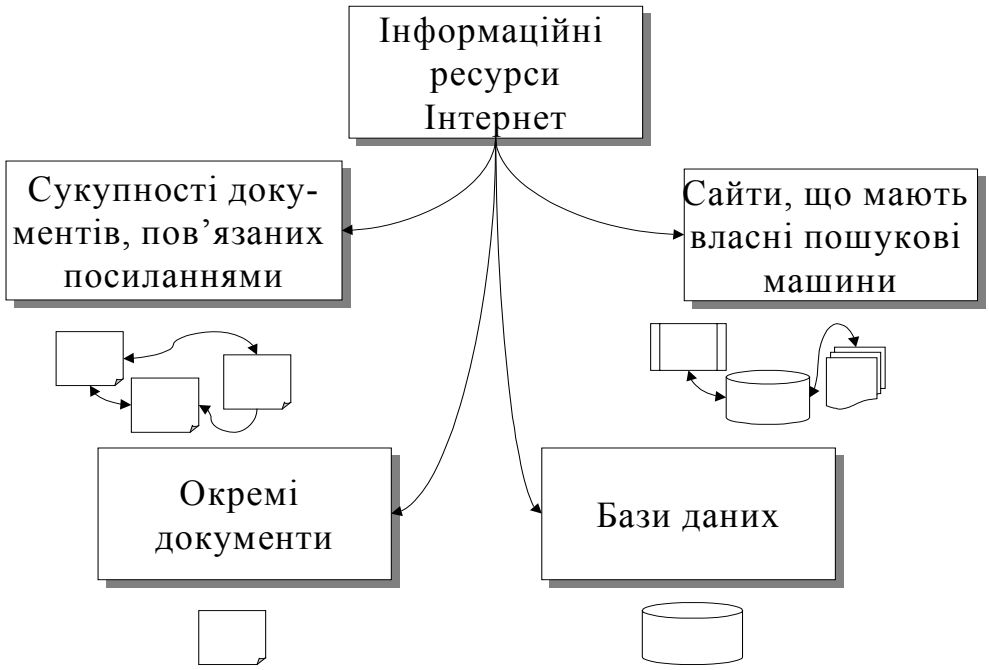


Рис.3.4. Типи зберігання ІР Інтернету

При збільшенні обсягу інформації на сайті її складність і вимоги до підтримки викликають необхідність зберігати інформацію в БД, яка враховує особливості ПрО сайту. Web використовується лише як універсальний інтерфейс до цієї БД. Значна частина інформації, наданої кінцевому користувачеві у форматі HTML, формується динамічно (у відповідь на дії користувача потрібні дані витягаються з БД і по них формується деякий текст).

Зараз обсяг «глибинної» частини Web (Deep Web) у багато разів більше «поверхневої» (Surface Web) [43], і це співвідношення

продовжує збільшуватися, оскільки тенденція до збереження інформації в структурованих джерелах очевидна і принаймні в найближчі роки не зміниться.

Локальний пошук по окремому Web-сервері можна організувати кількома способами. Якщо сервер змінюється досить часто, то краще використовувати локальний пошук за допомогою спеціалізованої пошукової машини, що встановлюється на Web-сервер і індексує тільки його. Приклади відомих продуктів, призначених для цього, – YandexSite компанії CompTek і «Слідопит» компанії MediaLingua.

Хоча вся інформація може бути знайдена відвідувачем такого сайту за допомогою локальної пошукової машини, але глобальні пошукові машина, не пристосовані для роботи з динамічним контентом, не здатні її проіндексувати, внаслідок чого потенційний користувач узагалі не звернеться до цього сайту.

Інший спосіб організації локального пошуку – пошукові ПА, які встановлюються на клієнтську машину й аналізують інформацію з Web-серверів. Вони працюють повільніше, але дозволяють точніше настроїти механізм пошуку.

3.2. Засоби формального подання метазнань про інформаційні ресурси Інтернету

Процес пошуку в Інтернеті

Ефективний пошук інформації в Інтернеті через збільшення обсягу і розосередження джерел стає складнішим. При цьому критичним є не стільки час пошуку, скільки відбір інформації, що релевантна запиту користувача.

Пошук в Інтернеті ускладнює різноманіття форматів подання інформації, стихійність організації IP і слабку виразність запитів. Динамічність інформаційного середовища Інтернету, що вимагає постійного відновлення відомостей про наявність і місце розташування інформації, також ускладнює задачу.

Процес пошуку в інформаційному середовищі будемо розглядати як послідовну взаємодію моделі інформації, яку прагне отримати користувач, з моделями інформаційних ресурсів.

Модель інформаційного середовища, в якому здійснюється процес інформаційного пошуку, включає модель IP, що описує властивості і правила роботи з IP, та модель засобів отримання інформації про середовище та впливу на нього.

Інформаційний пошук – процес співставлення запиту користувача з даними про ІР, відомими інформаційно-пошуковій системі (ІПС), до якої надійшов цей запит [117].

Ефективне виконання пошуку залежить від наступного:

- засобів подання запиту;
- засобів подання знань про ІР;
- засобів співставлення запиту та інформації про ІР;
- обсягу ІР, доступ до яких має ІПС.

Оцінка якості результатів пошуку необхідна для визначення ефективності роботи ІПС і для її порівняння з іншими системами. Ефективність виконання пошуку визначається тим, чи знайдені необхідні користувачеві ІР і наскільки багато знайдено ІР, що не відповідають запиту, часом виконання пошуку тощо. Згідно [353], ефективність ІПС визначається критеріями споживача, у першу чергу, повнотою і точністю пошуку, а також його швидкістю.

Крім цього, при оцінці ІПС враховується, з якими типами даних може працювати ІПС (приміром, чи може вона здійснювати пошук мультимедійних даних), у якій формі представляються результати пошуку (тільки посилання на ІР, анотація, інформація про розмір та час створення ІР, вказівки на ключові слова тощо) і який рівень підготовки користувачів необхідний для роботи в цій системі (тобто потрібно користувачеві освоювати формальну мову подання запитів чи досить ввести запит природною мовою).

Швидкість пошуку – це час між введенням запиту та отриманням відповіді на цей запит.

Точність пошуку визначається тим, яка частина інформації, що надається у відповідь на запит, є *релевантною*, тобто стосується цього запиту. Релевантність пошуку оцінюється через співвідношення між кількістю документів, що задовольняють користувача, тобто відповідають його запиту, і загальною кількістю знайдених у результаті пошуку документів. *Коефіцієнт релевантності* пошуку R_{search} – відношення кількості отриманих релевантних результатів Q_{user} до загальної кількості документів Q_{search} у відповіді ІПС: $R_{search} = Q_{user} / Q_{search}$.

Повнота пошуку характеризується співвідношенням між усією релевантною інформацією, що є в базі, і тією її частиною, що включена у відповідь.

Повнота оцінюється через співвідношення між кількістю релевантних документів, знайдених в результаті пошуку, та

загальною кількістю релевантних документів, до яких має доступ (потенційно) ІПС.

Коефіцієнт повноти пошуку C_{search} є відношення кількості отриманих релевантних результатів Q_{user} до Q_{all} – загальної кількості доступних ІПС документів, релевантних даному пошуковому запиту: $C_{search} = Q_{user} / Q_{all}$.

Коефіцієнт втрат інформації – параметр, зворотний коефіцієнту повноти пошуку: $D_{search} = 1 - C_{search}$. *Коефіцієнт пошукового шуму* – параметр, зворотний коефіцієнту релевантності пошуку: $N_{search} = 1 - R_{search}$.

В ідеальній ІПС R_{search}=1 та $C_{search} = 1$. У реальних ІПС коефіцієнт повноти пошуку може досягати значень 0.7-0.9, а коефіцієнт точності знаходиться в межах 0.1-1.0 [286]. Ефективність ІПС звичайно визначається середніми значеннями цих характеристик.

Між величинами коефіцієнтів повноти і точності пошуку існує зворотна залежність, параметри якої визначаються інформаційно-пошуковою мовою ІПС. Наприклад, у [353] ефективність ІПС визначається як сума нормованих оцінок повноти та релевантності пошуку: $E_{all} = R_{norm} + P_{norm}$, $R_{norm} = 1 - \frac{\sum_{i=1}^{n} r_i + \sum_{i=1}^{n} i}{n*(N-n)}$,

$P_{norm} = 1 - \frac{\sum_{i=1}^{n} \log r_i + \sum_{i=1}^{n} \log i}{\log(N!/n!*(N-n)!)}$, де N – кількість документів у системі; n – кількість релевантних документів у системі; i – номера цих релевантних документів; r_i – їхні ранги в отриманій вибірці.

При розгляді інформаційного пошуку в мережі Інтернету виникає ряд факторів, що обмежують застосовність таких характеристик ефективності ІПС:

- кількість IP, розміщених в Інтернеті, надзвичайно велика, але точно невідома;
- кількість релевантних документів теж надзвичайно велика, але точно невідома;
- значна частка документів, поданих в Інтернеті, дублюються;
- користувач має обмежені ресурси (зокрема, час) для вивчення результатів пошуку, тому при оцінці ефективності пошуку потрібно враховувати тільки ту частину результатів, що може бути вивчена користувачем;

- поняття релевантності нечітке (приміром, документ може тільки частково задовольняти інформаційну потребу користувача) і визначається користувачем.

У [286] оцінка якості результатів інформаційного пошуку визначається через результати пошуку і їх співставлення з усім масивом доступних у мережі документів, але не містить параметрів, що залежать від вибору ІПС та індексу документів.

Для вивчення результатів пошукового процесу користувач може використовувати тільки певну вибірку приблизно релевантних документів. Характеристики вибірки розділяють на прямі і непрямі. Прямі характеристики властиві самій вибірці і документам, що ввійшли в неї. Непрямі характеристики визначаються методом побудови вибірки і масивом документів, за яким вона побудована.

Найважливіші прямі характеристики – розмаїтість M, упорядкованість U і номер першого релевантного документа n_1; непрямі – кореляція між релевантністю і її формальною оцінкою σ.

Коефіцієнт точності вибірки має враховувати нечіткість поняття релевантності (на основі нечітких множин, лінгвістичних перемінних або багатозначної логіки).

Дана вибірка I з n документів $I = \{d_j\}, j = \overline{1,n}$, і користувач визначає релевантність $r_j, j = \overline{1,n}$, кожного документа $d_j, j = \overline{1,n}$, у балах (від 0 для нерелевантних документів до m балів для найбільш релевантного документа, $m = \max_{j=1}^{n}(r_j)$). Уся вибірка I одержує оцінку $r(I) = \sum_{j=1}^{n} r_j \le n * m$.

Точність вибірки – $R(I) = \frac{1}{n * m} \sum_{j=1}^{n} r_j$. Відповідно коефіцієнт пошукового шуму можна ввести аналогічно, як $N(I) = 1 - R(I)$.

Розмаїтість вибірки M(I) – кількість істотно різних тематичних кластерів документів. Пошуковий запит практично завжди недостатньо точно відображає інформаційну потребу. Чим більше розмаїтість вибірки, тим ймовірніше наявність у ній документів, що відповідають інформаційній потребі, або, принаймні, релевантніші за пошуковий запит. В другому випадку з'являється можливість коригувати опис інформаційної потреби за допомогою зворотного зв'язку. Зі збільшенням M(I) якість вибірки зростає. Проте ємність оперативної пам'яті людини складає 7 ± 2 одиниці, а вибір з більшого

кількості варіантів викликає ускладнення. Отже, бажано, щоб кількість кластерів не перевершувало цю величину.

Для визначення M потрібно задати деякий метод кластеризації. Можна використовувати критерій поділу документів по кластерах, що приводить до угруповання образів документів у векторному просторі уздовж напрямків від вектора запиту. Нехай маємо два документи A і B, для яких відома їхня релевантність пошуковому запиту r(A), r(B), і взаємна релевантність r(A,B), причому r(A)>r(B). Тоді документ B належить до того ж кластеру, що і документ A, якщо r(A,B)>r(B).

Інший важливий параметр – впорядкованість вибірки за змістовною релевантністю U(I). Формальна релевантність може істотно відрізнятися від значень змістовної релевантності, що визначається користувачем. Коефіцієнт упорядкованості – $U(I) = 1 - d/d_{max}$, де d – кількість перестановок двох сусідніх елементів, яка потрібна для досягнення упорядкування; d_{max} – максимальна кількість перестановок.

Залежно від того, наскільки вірно сформульований пошуковий запит, можливі кілька класів ситуацій [286]. Якщо запит точний, то ймовірно, що величини σ, U і n_1 близькі до одиниці. Якщо запит погано сформульований і далекий від реальної інформаційної потреби користувача, тоді релевантні документи не будуть знайдені, $\sigma \approx -1$, а значення n_1 більше кількості документів у вибірці.

Між цими класами ситуацій знаходиться третій, найбільш розповсюджений, – запит близький до інформаційної потреби, але описує її не точно. Деякі кластери вибірки більш релевантні і корекція пошукового запиту в цих напрямках має поліпшити результати пошуку. Значення M визначає ймовірність існування таких кластерів і можливість уточнення запиту.

Традиційні методи організації пошуку інформації можна поділити на три групи: індексного пошуку, статистичні та з використанням БЗ.

Індексний пошук орієнтований на роботу зі структурованими БД, в яких містяться відомості про ІР.

Статистичні методи базуються на розрахунку частоти входження слів в документ. Частота появи слів запиту в документах визначає їх релевантність. Лінгвістичні зв'язки між словами ці методи не враховують.

Системи, які використовують БЗ, здійснюють пошук інформації за допомогою зовнішніх апріорних знань (приміром, враховують

синонімію природної мови, таксономію термінів певної ПрО, граматичні правила синтаксичного аналізу речень тощо).

Найбільш розвинені можливості пошуку надають сьогодні системи пошуку за ключовими словами. Вони поділяються на дві групи: пошукові машини (ПМ) та каталоги.

Пошукові машини звичайно містять три компоненти (рис.3.5):

- програму індексації ІР (робота), що автоматично індексує сайти;
- БД (індекс);
- програму сканування, що за запитом знаходить відповідні ІР.

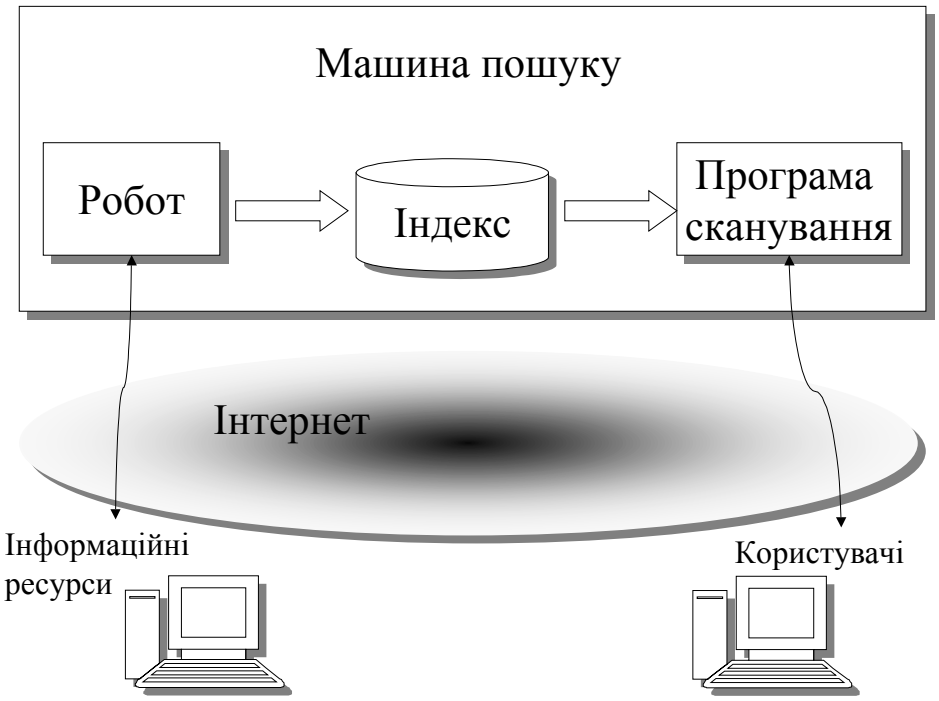


Рис.3.5. Архітектура машини пошуку

Роботи кожної пошукової системи намагаються самостійно проіндексувати всі ІР Інтернету. Деякі пошукові служби відносяться до повнотекстових – вони шукають ключові слова не тільки в заголовку і в метатегах ІР, але й у контенті документу, інші обмежуються заголовками і метатегами.

Різняться ПМ і за глибиною дослідження вузлів: одні обробляють тільки першу сторінку, інші – усі посилання до певного рівня, треті – весь Web-вузол. Крім того, деякі служби мають спеціалізацію (явну або неявну) і приділяють більше уваги вузлам, що присвячені певній темі. Найбільш відомі ПМ – AltaVista, HotBot, Яndex і Rambler.

У *каталогах* (рис.3.6) описи ІР, на відміну від ПМ, створюються не автоматично, а людьми-експертами, які вивчають нові вузли і відносять їх до відповідних тематичних категорій. Багато каталогів

також забезпечують пошук у своїй БД. Перевагою каталогів є більш висока якість опису IP, а недоліками – менший обсяг проіндексованих IP і неможливість автоматичного відновлення інформації про них. Найбільше відомі каталоги – Yahoo! і “Ау! “.

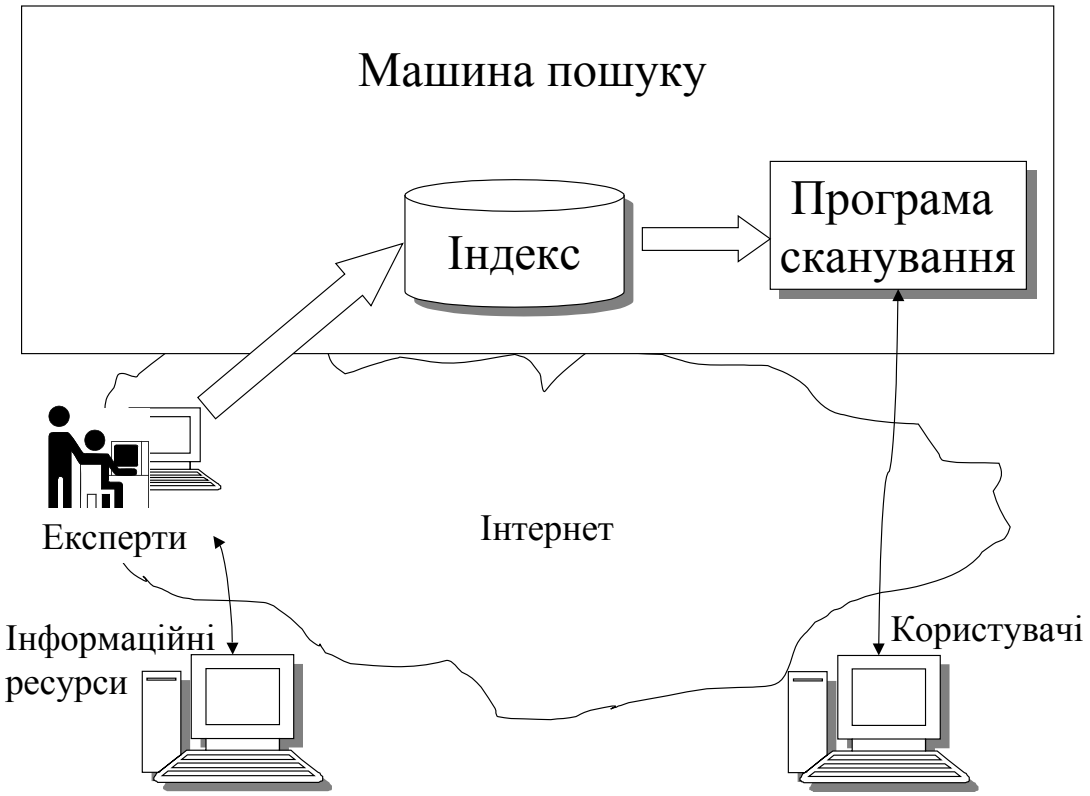


Рис.3.6. Архітектура каталога

Зараз розповсюджені як *локальні* ПМ і каталоги, що забезпечують пошук у рамках одного сайту, так і *глобальні* – для пошуку у всьому Інтернеті.

Суттєвим недоліком таких систем є низька точність інформації, що видається. ІПС використовують механізм пошуку по ключових словах і не враховують контекст, в якому існує інформація. Ось чому результатом роботи таких систем можуть бути сотні тисяч посилань.

Сучасні пошукові системи (приміром, Metacrawler, WebSeek) адресують запит користувача відразу до багатьох ПМ і складають індексні метакаталоги і БД. Але вони залишаються в рамках пошуку, основаного на ключових словах, тому отримані індекси зв'язують інформацію з термінами, враховуючи тільки актуальний для даного запиту лексичний або синтаксичний контекст. Аналогічні недоліки мають і тематичні каталоги. Крім того, що для їхнього створення і супроводу необхідно занадто багато часу, існує дисонанс між критеріями класифікації понять автора і користувачів.

Найбільший обсяг інформації про ІР Інтернету мають ПМ (точніше, вони мають інформацію про найбільшу кількість ІР, до яких можуть отримувати доступ інші користувачі глобальної мережі). ПМ можуть використовуватися іншими, більш розвинутими пошуковими засобами для доступу до цих ресурсів, тому для створення таких засобів потрібно знати їх можливості: які саме параметри інформаційних ресурсів накопичують в своїх БД різні ПМ і яку інформацію вони можуть надати у відповідь на запит.

У найпростішому випадку кожному посиланню на ІР приписується список ключових слів, за якими і надалі здійснюється пошук. Звичайно цей список містить тільки семантично значимі слова і не містить так звані стоп-слова (слова, що дуже часто зустрічаються в текстах). Списки ключових слів, побудовані різними ПМ для одного документа, можуть відрізнятися по двох причинах:

- різні списки стоп-слів;
- різні алгоритми обробки документів (пошук ключових слів тільки в заголовках або у всьому тексті; обробка тільки головної сторінки або усіх її посилань до визначеного рівня).

Крім списку ключових слів, для кожного ІР звичайно зберігаються його назва, розмір, дата створення і фрагмент тексту, що дозволяє судити про зміст сторінки. Варто мати на увазі, що ІР, посилання на який містить пошукова система, може вже не існувати. БД ПМ можуть містити й інші дані про ІР. Наприклад, у БД AltaVista є його індекс цитування – кількість посилань з інших сторінок, зареєстрованих у БД, на цей документ.

Переважна більшість ІПС – універсальні і не залежать від особливостей конкретних ПрО. Спеціалізовані ІПС враховують специфіку певної ПрО і тому надають релевантніші результати, але не можуть гарантувати виявлення усіх (або хоча б значної частини) тих інформаційних джерел, що належать до галузі їх спеціалізації і можуть бути виявлені універсальними ІПС.

Метапошукові системи

Метапошукові системи (МПС) не мають власної БД, але можуть використовувати БД інших ПМ. Вони не індексують самостійно ІР. Замість цього вони використовують БД інших ІПС, що значно облегшує формування індексу та дозволяє поєднувати інформацію, зібрану різними ІПС. Проте це накладає обмеження на параметри, за якими МПС можуть здійснювати пошук, що обумовлюються структурою БД тих ІПС, які вони використовують.

МПС має виділити найбільш релевантні документи [356]. Аналіз отриманих описів документів не здійснюється. Архітектуру такої системи подано на рис.3.7.

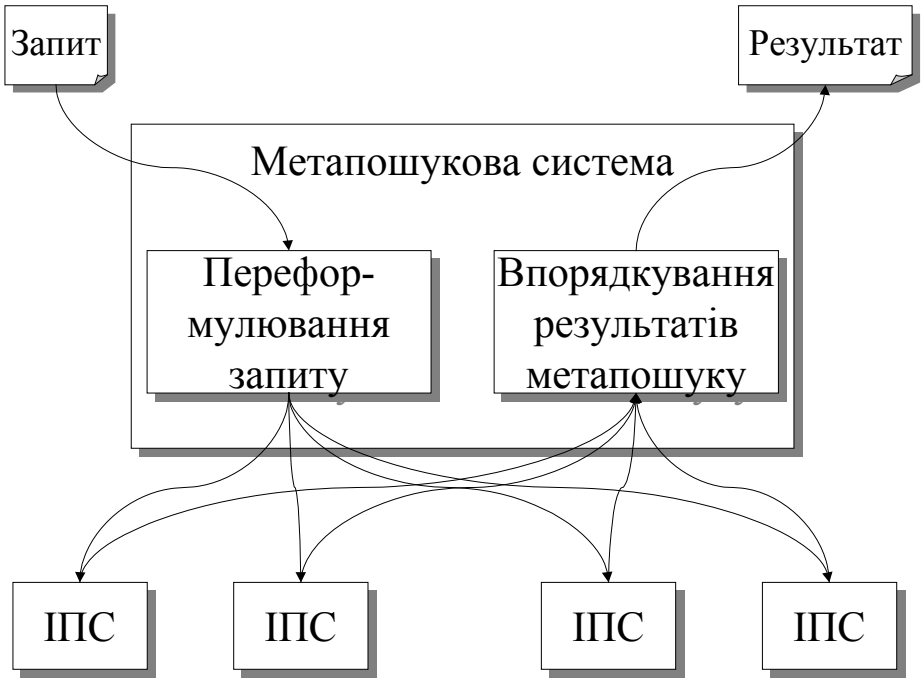


Рис.3.7. Архітектура метапошукової системи

При розробці наступного покоління МПС враховані можливістю вибору тих ПМ, у яких, на думку користувача, з більшою імовірністю можна знайти потрібну інформацію (рис.3.8).

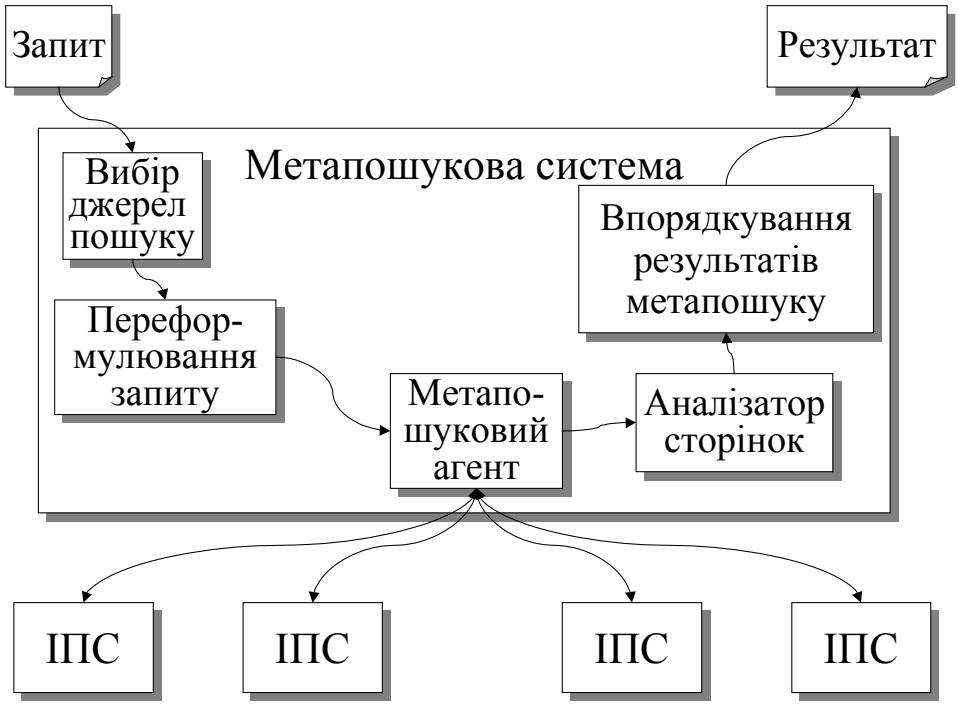


Рис.3.8. Архітектура метапошукової системи другого покоління .

Такий підхід дозволяє зменшити обчислювальні ресурси метапошукового серверу, не перевантажуючи його великим обсягом непотрібної інформації. Слід відзначити, що в будь-якій МПС найбільш вузьким місцем є пропускна здатність каналу передачі даних, тому що обробка сторінок з результатами пошуку, отриманими від кількох десятків пошукових серверів не є занадто складною операцією, оскільки витрати часу на обробку інформації набагато менші часу очікування сторінок, запитаних у пошукових серверів. Як приклад систем, що мають подібну організацію, можна назвати Profusion, Ixquick, SavvySearch, Metaping.

Метапошукова система (МПС) – пошуковий інструмент, що посилає запит одночасно до кількох ІПС (машин пошуку, каталогів або локальних пошукових механізмів, які забезпечують доступ до невидимої частини Інтернету).

Можна виділити чотири типи МПС [294]:

- "*справжні*" МПС, що поєднують і впорядковують результати пошуку, отримані від різних ІПС, на одній сторінці;
- псевдо-МПС *першого* типу, що групують результати за ІПС, але розташовують ці результати на одній сторінці;
- псевдо-МПС *другого* типу, що відкривають для кожної використаної ІПС окреме вікно та розташовують результати пошуку, отримані від кожної з ІПС, на окремій сторінці;
- *пошукові утиліти* – програмні засоби, які здійснюють пошук у кількох ІПС.

Прикладами "справжніх" МПС є системи ez2www, Vivisimo, Query Server, Infonetware, Iboogie, Fazzle.

До псевдо-МПС першого типу можна віднести Mall Agent, qb Search, Better Brain, My Net Crawler, Planet Search, Rede Search.

Псевдо-МПС другого типу можна поділити на дві категорії. У першому випадку запит формується тільки один раз, потім обираються ІПС, до яких цей запит спрямовується. Для кожної ІПС відкривається нове вікно. Прикладами таких систем є Multi-Search-Engine, GoGettem, Search Bridge, Net Depot. В іншому випадку спочатку обирається ІПС, будується запит у форматі цієї ІПС та відкривається відповідне вікно. До таких МПС відносяться Alpha Seek, Westlaser, Dan's No Overhead Search Thingy, Express Find Freelity. Прикладами пошукових утиліт є BullsEye, Copernic, Lexibot, WebFerrer, WolfBot.

Для оцінки якості функціонування МПС використовують як спільні з іншими ІПС критерії – кількість і релевантність результатів, здатність використовувати додаткові способи пошуку, надання користувачам можливості визначати свої умови, швидкість пошуку тощо, так і специфічні – кількість ІПС, до яких звертається ІПС, можливість підключення користувачем нових ІПС, зручність керування групами ІПС, до яких спрямовується запит.

Найбільш важливими критеріями оцінки є повнота та точність пошуку. Повнота пошуку – ступінь охоплення інформаційних джерел, що можуть містити інформацію, що цікавить користувача. Точність – ступінь релевантності знайдених по запиті користувача документів. Точність пошуку можна підвищити за рахунок додаткової обробки і семантичної фільтрації знайдених документів.

Методи підвищення точності пошуку поділяються на лінгвістичні та нелінгвістичні.

Пошук тільки за ключовими словами часто не відповідає реальним потребам користувача. Можна розглядати запит не як послідовність слів, а як зв'язний текст та аналізувати його за законами мови. Приміром, в МПС «Сиріус» висока точність пошуку досягається в результаті використання методу семантико-синтаксичного аналізу, заснованого на принципах комунікативно-граматичної школи [290] і методах опису концептуальної системи предметної області [168].

Ключове слово в запиті являє собою не лексему, тобто одиницю словникового складу мови в сукупності його конкретних граматичних форм і не є синтаксичною одиницею. У зв'язному тексті слово несе тільки свій індивідуально-лексичний, але й узагальнений, категоріальний зміст у конструкціях різного ступеня складності. Ці одиниці – синтаксеми – характеризуються взаємодією морфологічних, семантичних і функціональних ознак. Тому у процесі пошуку потрібно обробляти саме синтаксеми, тобто не тільки лексичне, але і похідне від нього синтаксичне значення компонента запиту.

Нелінгвістичні методи враховують відносну значимість різних структурних фрагментів тексту. Розглядають два типи значимості: статичну (значимість фрагмента як деякої структурної одиниці тексту залежно від його типу – заголовок, підзаголовок тощо) і динамічну (значимість для конкретного запиту). В процесі впорядкування документів, що є результатом пошуку, враховують значимість фрагментів тексту документів, у яких знайдені ключові слова або інші релевантні запиту структури. Документ у цьому випадку

представляється як набір текстових фрагментів різного типу, причому множина типів фрагментів упорядкована. Кожен тип фрагмента має вагу, яка враховується при підрахунку результуючої релевантності. Наприклад, документи, що містять ключові слова у назві, будуть більш значимі для користувача порівняно з тими, що не мають ключових слів у назві документами, при однаковій релевантності по всіх типах.

Повноту пошуку в МПС можна підвищити шляхом розширення пошукового запиту синонімами та конверсивами (якщо відповідне дієслово є членом конверсивної пари – наприклад, "купити-продати") ключових слів. Ця процедура передбачає наявність словників синонімів та конверсивів.

Засоби подання інформаційних запитів користувача

Запит користувача – це опис ІР, доступ до яких він прагне отримати. Такий запит може, приміром, містити ключові слова, що пов'язані логічними операторами, або вказувати тип ІР і його тему за класифікатором, обмежити його розмір та дату створення.

У загальному випадку інформаційний запит

$Z = \langle Z_p, O_p, P_p, M_p, S_p, R_p, K_p, n, t \rangle$, де:

- Z_p – ключове слово (словосполучення) чи кілька ключових слів, які мають бути присутні в ІР, що пов'язані логічними операторами:

$Z_p = x_{l_1} \wedge ... \wedge x_{l_n} \vee ... \vee x_{k_1} \wedge ... \wedge x_{k_m}$, де $x_{i_j}, i = \overline{1,k}, j = \overline{1,m_k}$, – слова чи словосполучення природної мови чи їхні заперечення (приміром, x_1="програмний агент", x_2=¬"мультиагентна система“). Кількість та синтаксичні правила використання логічних операторів специфічні для конкретних ІПС;

- O_p – контекст запиту, що створюється шляхом аналізу дій користувача автоматично або безпосередньо користувачем (приміром, на основі онтології ПрО) – слова або словосполучення, що пов'язані логічними операторами:

$O_p = t_{l_1} \wedge ... \wedge t_{l_n} \vee ... \vee t_{k_1} \wedge ... \wedge t_{k_m}$, де $t_{i_j}, i = \overline{1,k}, j = \overline{1,m_k}$, – слова чи словосполучення природної мови чи їхні заперечення;

- P_p – параметри ІР (розмір, дата створення, мова документа, формат подання тощо);
- M_p – ІР, що забезпечують доступ до інших ресурсів (приміром, пошукові машини, за допомогою яких слід здійснювати пошук, або

посилання на ІР, які потрібно перевірити на відповідність іншим параметрам запиту);

- S_p – статус запиту (постійний чи одноразовий);
- R_p – пріоритет запиту;
- $К_p$ – класи посилань, що цікавлять користувача;
- n – максимальна кількість ІР, які хоче отримати користувач;
- t – час, за який слід виконати пошук.

Слід зазначити, що в інформаційному запиті практично будь-які параметри можуть бути необов'язковими, але завжди має бути визначений хоча б один з них.

На жаль, більшість існуючих ІПС не зберігає інформацію про попередні сеанси роботи з користувачем, не має відомостей про його ставлення до конкретного ІР і тому пропонує користувачу аналізувати усі знайдені за запитом посилання знов кожного разу.

Для пошуку в Інтернеті користувачі зазвичай задають запит до ІПС з 2-3 ключових слів. Виконання таких запитів вкрай не ефективне. Основні проблеми пов’язані не з часом виконання запиту або відсутністю інформації про необхідні ІР, а з великим обсягом нерелевантних посилань, що відповідають запиту, але не задовольняють справжні інформаційні потреби користувача.

Ускладнення форми подання запиту підвищує релевантність пошуку, але призводить до збільшення часу пошуку через більш складну процедури обробки. Крім того, використання складніших форм подання запиту, навіть якщо такі можливості підтримуються в ІПС, недоступне більшості користувачів через відсутність спеціальних знань.

Найбільш розповсюджене подання документів в сучасних ІПС базується на векторній моделі – документ подається як набір термінів, що відображають його зміст. Інформаційно-пошукова мова (ІПМ) дозволяє будувати з термінів логічні вирази, використовуючи булеві оператори AND, OR, NOT. Така схема досить проста, однак має значні недоліки: оператор AND може дуже сильно зменшити множину документів, що відповідають запиту, а оператор OR – призвести до невиправданого його розширення. Для успішного застосування цієї ІПМ потрібно добре знати лексику ІР.

Існують ІПМ, що не використовують логічні оператори. Для задання запиту може, наприклад, застосовуватися документ-зразок. Найпростішою моделлю типу "Like this" є лінійна модель індексування і пошуку, за якою близькість документа і запиту розглядається як кут між ними. У цьому випадку обчислюється синус

кута, який є скалярним добутком двох векторів. Документи ранжуються відповідно до значень міри близькості.

Така міра близькості не зовсім придатна для пошуку IP в Інтернеті, тому що запити мають велику довжину. Тому реально застосовуються інші міри близькості, але принцип залишається той же: спочатку обчислюється міра, а потім відбувається ранжирування.

Досить ефективними є ІПМ, що базуються на нечітких множинах. При даному типі пошуку весь масив документів описується як набір нечітких множин термінів. Кожен термін визначає певну монотонну функцію приналежності до документів даного масиву. Оператор AND складається як мінімум із двох функцій, що відповідають термінам запитів: OR – як максимум, NOT – як 1-<значення функції>. Результати пошуку впорядковуються відповідно до отриманих значень. Цей метод використовується тільки в дослідницьких системах, його поширення вкрай обмежене.

Сучасні ІПС Інтернету мають індексні БД величезного обсягу. Впорядковувати таку інформацію цілком практично неможливо. Тому застосовуються моделі, що задають граничні значення для документів, які пропонуються користувачеві.

У *кластерній* моделі можуть використовуватися два підходи:

- масив заздалегідь розбивається на підмножини документів і при пошуку обчислюється близькість до певної підмножини;
- кластер будується навколо певного запиту і найближчих до нього термінів.

Найчастіше ця модель застосовується в системах, що уточнюють запит за релевантністю знайдених документів.

При *імовірнісній* моделі обчислюється ймовірність приналежності документа класу релевантних запиту документів з урахуванням термінів запиту для кожному з документів БД.

Проаналізуємо ІПМ, які використовуються в найпопулярніших машинах пошуку Інтернет.

Lycos дає можливість застосовувати простий запит і більш складний метод пошуку. У простому запиті як пошуковий критерій вводиться речення природною мовою, після чого Lycos нормалізує запит, видаляючи з нього стоп-слова, і тільки після цього переходить до його виконання. Видається інформація про кількість документів, що відповідають кожному слову, а потім – список посилань на формально релевантні документи. Для кожного документа вказується міра його близькості до запиту та кількість слів із запиту, що

містяться в документі. У розширеній формі запиту можна використовувати логічні оператори.

Найцікавіша можливість *AltaVista* – це розширений пошук. На відміну від багатьох інших систем вона підтримує унарний оператор NOT. Оператор NEAR реалізує можливість контекстного пошуку – терміни мають розташовуватися поруч у тексті документа. AltaVista підтримує пошук по ключових фразах, при цьому вона має досить великий фразеологічний словник. Крім того, при пошуку в AltaVista можна задати ім'я поля, де має зустрітися слово: гіпертекстове посилання, applet, назва образу, заголовок тощо.

В ІПМ *Yahoo* всі слова потрібно вводити через пробіл, вони поєднуються зв'язуванням AND або OR. При видачі результату не вказується ступінь відповідності документа запиту, а тільки підкреслюються слова запиту, що містяться в документі. Нормалізація лексики не здійснюється, не проводиться аналіз на "загальні" слова. Ранжирування здійснюється за кількістю термінів запиту в документі.

Система *Infoseek* має досить розвинуту ІПМ, що дозволяє не просто вказувати, які терміни мають зустрічатися в документах, але і зважувати їх. Досягається це за допомогою спеціальних знаків: "+" – термін повинний бути в документі; "-" – термін має бути відсутнім. Крім цього, Infoseek дозволяє проводити контекстний пошук. Це значить, що, використовуючи спеціальну форму запиту, можна визначити, які слова мають знаходитися у документі поруч.

Методи спрямованого кроулінгу

Перспективний напрямок пошуку в Інтернеті – *спрямований кроулінг*, який має значні відмінності від звичайного пошуку за допомогою ПМ та каталогів. Він призначений для дослідження гіпертекстових ресурсів шляхом цілеспрямованого пошуку сторінок, що релевантні певним темам, які цікавлять користувача. Для специфікації цих тем використовуються не ключові слова, а зразки документів. ІПС, що базуються на спрямованому кроулінгу (приміром, Focused Crawler [33]), не охоплюють всі доступні через Інтернет IP, а спрямовані на пошук меж, що відокремлюють посилання у наданих ним зразках, які доцільно перевірити на релевантність запиту, від “небажаних” районів Інтернету, які слід уникати. Це дозволяє заощаджувати час в процесі пошуку.

Найголовніше в стратегії розробки системи спрямованого кроулінгу – забезпечити *релевантність* і *якість* знайдених

документів, а не їх обсяг. У ході ряду експериментів було виявлено, що пошук за ключовими словами не призводить до ефективного виявлення ресурсів, що відносяться до досить специфічних ПрО. Також уявляється недоцільним переглядати й індексувати сотні мільйонів сторінок Інтернету тільки для того, щоб знайти кілька десятків ресурсів, потрібних користувачеві.

Перші Web-кроулери просто рухалися по кожному посиланню, виявляючи нові сторінки. Кроулери й агенти більш складні. Самим раннім прикладом використання запитів для спрямованого кроулінгу є система Fish Search. Про схожі результати повідомляють WebCrawler [36], Shark Search [90]. Спрямований кроулінг відрізняється використанням таксономій тем, навчанням по прикладах і застосуванням дистиляції графів для відстеження концентраторів тем.

Спрямовані кроулери використовують аналіз соціальної мережі, який вже протягом десятиліть намагається знайти вузли з високим авторитетом. Але існує ряд важливих відмінностей у тому, як роблять це різні системи. PageRank не розглядає контент сторінки, а HITS і CLEVER досліджують Web у фіксованому радіусі від відповіді на запит по ключових словах. У спрямованому кроулінгу не існує заданого заздалегідь радіуса дослідження, тому що він за допомогою дистилятора і класифікатора визначає його сам.

Для цілеспрямованого кроулінгу потрібно мати дві програми дослідження гіпертексту:

- *класифікатор* оцінює релевантності гіпертекстового документа відповідним темам;
- *дистилятор* знаходить вузли гіпертексту, що мають доступ до великої кількості релевантних сторінок через кілька посилань.

Системи спрямованого кроулінгу забезпечують пошук, індексування й обслуговування сторінок, що відносяться до певного набору тем, які представляють відносно вузький сегмент Інтернету. Таким чином, уміст Web обробляється розподіленою командою кроулерів, кожний з який спеціалізується на певних темах. Кожний з них набагато гнучкіше аналізує сторінки за темами, на які він націлений. Під керівництвом класифікатора спрямований кроулер вчиться визначати релевантність за прикладами, вбудованими у таксономію тем.

Ціль розробки спрямованих кроулерів – створення адекватної структури тем для всього Web, що забезпечує засоби виконання

частково структурованих запитів користувачів і їх аналізу. Для систем спрямованого кроулінгу потрібно здійснювати:

- дослідження соціології посилань;
- виявлення спеціалізації сайтів;
- навчання темам під керуванням людини, що забезпечує добру фільтрацію, але вимагає великих витрат праці;
- визначення культури співтовариства: проста статистика про граф посилань (link graph) показує важливу інформацію про співтовариство, що цікавиться цією темою, наприклад, за який саме час добрий ресурс здобуває популярність;
- оцінку масштабу часу співтовариства (Estimating community timescales): прості запити дозволяють ідентифікувати тематичні регіони Web, що швидко зростають, або, навпаки, відносно стабільні.

Для досвідчених користувачів Web більш корисні спрямовані канали (focused portholes), чим загальні портали [86]. Спрямовані кроулери являють собою побудовані на прикладах (example-driven) автоматичні генератори таких каналів.

Адміністрування кроулерів

Основою спрямованого кроулера є канонічна таксономія тем із прикладами. Користувач має вибрати або визначити вузли таксономії тем, або вказати адреси (URL) прикладів, що послужать стартовими точками для кроулінгу.

Огляд операцій спрямованого кроулінгу:

- *створення канонічної таксономії* (Canonical taxonomy creation): при створенні системи класифікатор попередньо тренується з канонічною таксономією (такою, наприклад, як Yahoo!, The Open Directory Project, The Virtual Library or The Mining Co.) та з відповідним набором прикладів (дерево канонічної класифікації є частиною початкової системи);
- *набір прикладів* (Example collection): користувач збирає адреси URL прикладів, що його цікавлять, вони розглядаються системою, і з них імпортуються закладки;
- *вибір і очищення таксономій* (Taxonomy selection and refinement): система пропонує найбільш загальні класи, а користувач може вибрати деякі з них і відзначити як добрі (якщо користувач вважає отриману таксономію надто грубою, він може очистити деякі категорії або перемістити документи з однієї категорії до іншої);
- *інтерактивні дослідження* (Interactive exploration): система пропонує користувачеві включити в набір прикладів деякі з

додаткових URL-адрес, що знаходяться близько від прикладів та можуть бути схожі на них;

- *навчання* (Training): класифікатор інтегрує удосконалення, зроблені користувачем у моделях статистичних класів;
- *дослідження ресурсів* (Resource discovery): на цьому етапі система готова здійснити дослідження ресурсів;
- *дистиляція* (Distillation): система використовує алгоритм дистиляції тем для ідентифікації сторінок, які містять велику кількість посилань на релевантні ресурси: концентратори, "добірки", переліки посилань (hubs);
- *дослідження пов'язаних класів* (Discovery of related classes): структура таксономії допомагає користувачеві знаходити райони Web, що відповідають його інтересам, але не пов'язані зі стартовим набором документів;
- *зворотний зв'язок* (Feedback): система повідомляє користувача про популярні сайти, а користувач вирішує, потрібні вони йому чи ні.

Користувач може класифікувати шлях до конкретної сторінки, відмітивши всі вузли. В іншому варіанті можна обробляти кроулером тільки ті приклади, що знайдені користувачем, і не застосовувати створені раніше таксономії. Таким чином, одержуємо просту задачу навчання для двох результуючих класів (“релевантний” – “нерелевантний”) для кожної цільової теми. Такий підхід має кілька переваг: 1) краще моделюються негативні класи; 2) можна повторно використовувати тренувальні спроби навчання: працювати з описом інтересів користувача через раніше визначений набір категорій і його очищення значно простіше, ніж знаходження адекватної кількості позитивних і негативних прикладів.

Архітектура систем спрямованого кроулінгу

Кожна система спрямованого кроулінгу зазвичай складається з трьох основних компонентів (рис.3.9).:

- класифікатора;
- дистилятора;
- кроулера.

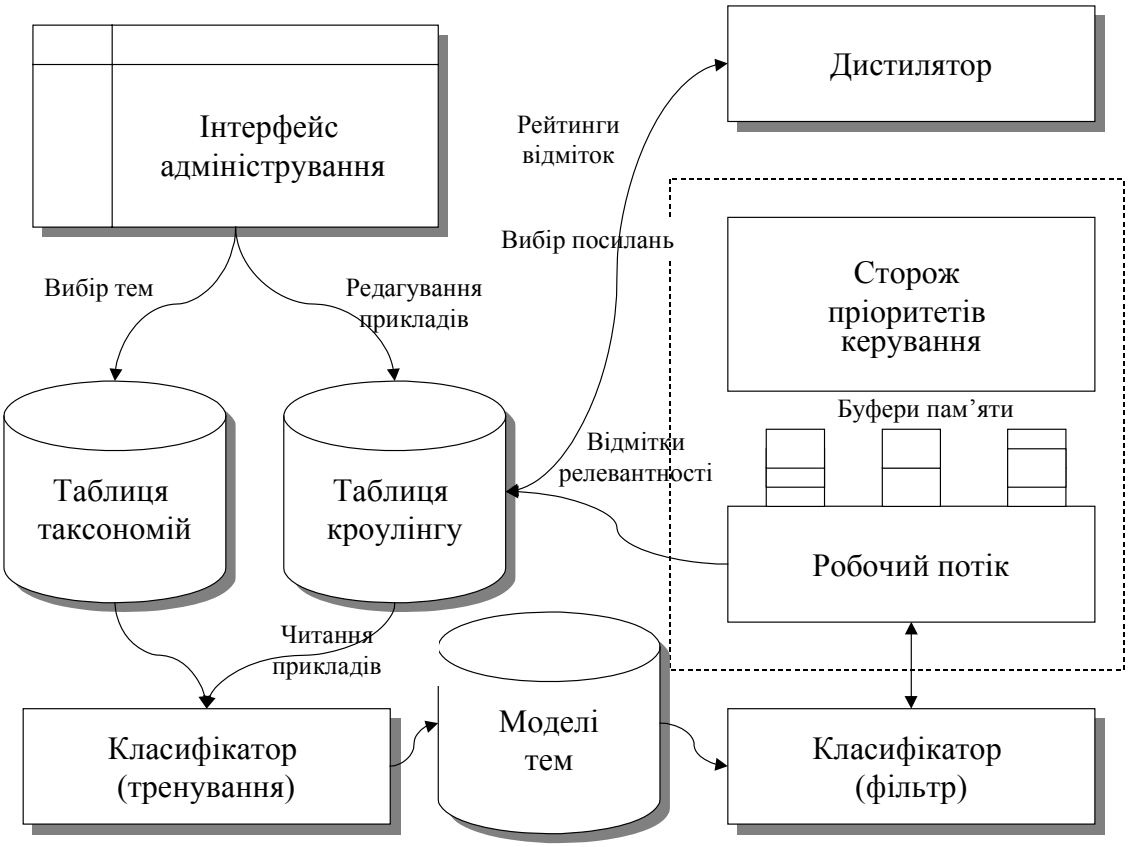


Рис.3.9. Архітектура систем спрямованого кроулінгу

Розглянемо спрямований гіпертекстовий граф G, вузли якого фізично розосереджені. У даному випадку G – це Web. Відома вартість відвідування кожної його вершини – сторінки Web. Є також деревоподібний ієрархічний каталог тем C, подібний Yahoo!. Кожен вузол теми $c \in C$ пов'язаний з певними сторінками з G як із прикладами. Визначимо множину прикладів, пов'язаних з темою c, як D(c). Ці сторінки можуть розглядатися як "бажані" для системи.

Інтереси користувача характеризуються підмножиною тем $C^* \subset C$, відзначених як добрі. Жодна така тема не є предком іншої доброї теми. Предки доброї теми називають маршрутами.

Потрібно задати сторінку Web q і міру її релевантності $R_{C^*}(q)$ відносно C^*, а також метод обчислення релевантності. Можна використовувати ймовірнісну міру $0 \le R(q) \le 1$. За визначенням, $R_{root}(q) = 1 \forall q$. Якщо $\{c_i\}$ є нащадками c_0, тоді $\sum_{c_i} R_{c_i}(q) = R_{c_0}(q)$.

Система починає роботу з відвідування всіх сторінок з $D(C^*)$. На кожному кроці вона перевіряє множину V сторінок, що відповідають гіперпосиланням на одній чи кількох відвіданих раніше сторінках. Неформально метою є відвідати якнайбільше релевантних сторінок і

якнайменше – не релевантних. Тому потрібно знайти $V \supseteq D(C^*)$, де V досяжно з $D(C^*)$ і $\sum_v R(v)/|V| = \max$.

Таке формулювання може призвести до безвихідної проблеми, якщо сторінки з усіх тем розосереджені по всьому Web.

Однак таке зустрічається рідко. Цитування означає навмисну оцінку автора сторінки. Хоча частина цитувань зашумлена, більшість цитувань пов'язані із семантично близькими матеріалами. Таким чином, релевантність сторінки є значимим індикатором релевантності її сусідів, причому вірогідність цього правила падає зі збільшенням радіуса сусідства. Це пояснює необхідність використання класифікатора. Крім того, велика кількість цитувань з одного документа пов'язана з підвищенням релевантності документів, що цитуються. Тому дистилятор використовується для ідентифікації сторінок, що мають велику кількість посилань на релевантні сторінки. Релевантність забезпечується кроулером за допомогою використання класифікатора гіпертексту.

Передбачається, що таксономія категорій містить ієрархічний розподіл документів Web. У реальному житті документи досить часто належать одночасно до кількох категорій, але в даній моделі це не враховується. Категорії дерева таксономії, які називають вузлами, позначаються як c. good(c) – вузол, позначений як "добрий".

За визначенням, для кожного документа d імовірність того, що він буде згенерований з кореневого каталогу, дорівнює 1. У загальному випадку $\Pr[c \mid d, parent(c)] = \frac{\Pr[c \mid d, parent(c)]\Pr[d \mid c]}{\sum_{c`parent(c`)=parent(c)} \Pr[d \mid c`]}$, якщо сума ранжується для всіх нащадків c` від c.

Щоб знайти $\Pr[d \mid c]$, необхідна для генерації документів. $\Pr[c \mid parent(c)]$ визначає основний розподіл документів. Генератор сторінок спочатку визначає тему, до якої слід приписати документ d, використовуючи його імовірності для вибору вузла – листа c^*.

Кожен клас, зокрема, c^* може мати вид, який містить будь-які терміни. $\theta(c^*, t)$ – ймовірність виду t. Генератор знаходить довільну довжину $n(d)$ для документа. Потім він багаторазово виконується для c^* і записує терміни, що відповідають виду. Таким чином, документ являє собою набір слів (без інформації про їхній порядок і зв'язки між ними). Якщо термін t зустрічається $n(d,t)$ раз, то $\Pr[d \mid c] = \binom{n(d)}{\{n(d,t)\}} \prod_{t \in d} \theta(c,t)^{n(d,t)}$.

Незважаючи на свою простоту, ця модель виявилася дуже вдалою. У процесі кроулінгу, знайшовши документ, ми прагнемо знайти найкращий клас – лист c^*. Для класифікації можна застосувати дві моделі фокусування.

1. Сильне правило фокусування (Hard focus rule): якщо предок c^* відзначений як добрий, то надалі дозволяється відвідувати URL-адреси, знайдені в документі d, інакше кроулер видаляє d.

2. Слабке правило фокусування (Soft focus rule): ймовірність того, що сторінка релевантна, визначається як $R(d) = \sum_{good(c)} Pr[c \mid d]$, тому що жоден добрий вузол не є предком іншого.

Пріоритет відвідування кожного сусіда поточної сторінки d дорівнює релевантності d. У випадку, якщо до сторінки веде кілька шляхів, вибирають той, що має максимальну релевантність.

Релевантність не є єдиним атрибутом, що використовується для оцінки сторінки в процесі кроулінгу. Великий документ, високо релевантний темі пошуку, який не містить посилань, є тільки кінцевою точкою кроулінгу. Вдалою стратегією для кроулінгу є ідентифікація концентраторів (hubs) – сторінок, що являють собою набір посилань на авторитетні інформаційні ресурси, релевантні темі. Аналіз соціальної мережі призначений для розгляду властивостей графів, що виникають між такими сутностями, як люди, організації, статті тощо через співавторство, цитування, керівництво, оплату, телефонію і багато іншого.

Авторитет (Prestige) – це важливий атрибут вузлів соціальної мережі. Кількість цитувань статті u є обґрунтованою, але сирою мірою авторитету $p(u)$. Кращою мірою є вагове цитування, або загальний авторитет видань, які цитують дану статтю. Таке визначення є циклічним, але може бути дозволено ітеративними обчисленнями для перебування фіксованої точки $p = Ep$, де E – спрямована матриця суміжності (adjacency).

Існують два типи вузлів:

- вузли з високою централізацією;
- концентратори, пов'язані з авторитетами.

Кожен вузол v має два відповідних рахунки: $h(v)$ і $a(v)$. Потім наступні ітерації повторюються на граничному наборі E потрібну кількість разів: $a(v) \leftarrow \sum_{(u,v)\in E} h(u)$ та $h(u) \leftarrow \sum_{(u,v)\in E} a(v)$.

Кроулер має один сторожовий потік (watchdog thread) і багато робочих потоків (worker threads). Сторож перевіряє роботу на границях кроулінгу. Робочі потоки зберігають інформацію про нові

досліджені сторінки на свої власні робочі диски. Потім ці результати накопичуються в загальній БД. Класифікатор викликається кожним робочим потоком при виявленні нової сторінки. Значення R обчислюється як частина результату інформації про сторінку.

Існує багато інших індикаторів роботи кроулерів. Найбільш важливі з них – *релевантність*, *сфера дії* (coverage) і *якість дослідження ресурсів*.

Навігація в таксономіях

Для ефективного виконання спрямованого кроулінгу потрібно мати відповідні засоби здійснення пошуку і навігації в таксономіях тем. Проблема полягає в тому, щоб побудувати систему, здатну задовольняти кільком умовам:

- документи, завантажені в БД, мають бути проіндексовані окремо від ключових слів шляхами тем у таксономії, для чого потрібний надійний автоматичний ієрархічний класифікатор, що дозволяє враховувати специфіку термінології;
- таксономія може використовуватися для подання користувачеві очищених наборів документів, отриманих у відповідь на запит;
- система має бути швидкою, особливо якщо вона використовується разом з кроулером або службою розсилання новин;
- система має ефективно поновлювати свої знання.

Приміром такої автоматизованої системи, що здійснює пошук і навігацію в таксономіях, є *TAPER* [34] (taxonomy-and-path-enhanced-retrieval). Для кожного вузла таксономії вона виділяє терміни характерних рис (*feature)* і шуму (*noise)*, обчислюючи найкращі дискримінанти для цього вузла. Для класифікації нового документа використовуються тільки характерні терміни, яких не дуже багато. Тому моделі класів не великі і класифікація здійснюється швидко. На відміну від інших класифікаторів, що використовують плоску множину (flat set) класів, набір характерних термінів змінюється залежно від контексту в процесі руху документа по таксономії. Це дозволяє відфільтровувати загальний жаргон на кожному кроці. Моделі тексту, побудовані в кожному вузлі, також надають засоби підсумовування кількості документів, що використовують кілька дескриптивних ключових слів, – сигнатуру.

Високоточна ідентифікація тем необхідна в двох контекстах: запитів і фільтрації. Більшість запитів до пошукових машин дуже короткі. З такими запитами зв'язана проблема надмірності (*abundance)*, тому що існує багато різних інтерпретацій типів ключових слів. При фільтрації потік документів створюється в режимі

on-line. Система збирає профілі інтересів користувача і використовує їх для фільтрації. У найпростішій формі профілем може бути набір термінів і фраз, явно заданий користувачем. Більш корисно розглядати як профіль набір документів, що користувач переглянув і які йому сподобалися. Це детально характеризує користувача і добре працює для невеликих систем, але в масштабах Web система, що накопичує таку кількість деталей, неможлива.

Альтернативою є опис профілів не на рівні окремих документів, а на рівні вузьких стандартизованих тем, як у таксономії Yahoo!. Підхід застосовується в системі Surf Advisor.

Таким чином, текстовий пошук має базуватися на контексті теми і на ключових словах. Якщо відома загальна тема документа, він може бути охарактеризований термінами, що часто зустрічаються в моделі тем. Щоб ідентифікувати тему документа, автоматичний класифікатор має відділити в ньому слова, що містять інформацію про тему, від шуму.

Системи пошуку та навігації в таксономіях базуються на розробках в галузі дослідження даних, машинного навчання, повнотекстових БД і розпізнавання образів, статистичної теорії рішень (як класичної, так і байєсової).

Класифікатори можуть бути як параметричними, так і непараметричними. Найвідоміші типи непараметричних класифікаторів – дерева рішень (приміром, CART і C4.5) і нейронні мережі. Фундаментальні ідеї з галузі пошуку інформації, що використовувалися при розробці цієї системи, пов'язані з обробкою інформації на синтаксичному рівні (фільтрація стоп-слів, постачання позначками, морфологічний пошук, обчислення ваг термінів, виявлення відповідностей між документами і запитами).

Індексування повнотекстових документів

Для того щоб мати доступ до вмісту повнотекстового файлу, необхідно здійснити в ньому послідовний пошук, а це вимагає великих витрат часу. Одним з рішень проблеми є створення індексного файлу (індексу) документа, що відображає його вміст [255]. Індекс файлу складається зі списку елементів – значень ідентифікуючої властивості запису, та вказівок про місце розташування даного запису. Іноді зручно створювати індекс, що вказує тільки приблизне, а не точне місце розташування потрібного запису. Інший підхід до індексації файлів – створення індексу за ієрархічним принципом, коли загальний індекс файлу являє собою

багаторівневу або деревоподібну структуру (прикладом такої індексної структури є ієрархічна система каталогів, яку використовує більшістю операційних систем для роботи з файлами – каталоги виконують роль індексів, кожний з який містить посилання на свої підкаталоги, а вся файлова система може розглядатися як один великий індексований файл).

Індексування повнотекстових документів за допомогою ключових слів є найпоширенішою зараз технологією. Суть її полягає в тому, що для кожного документа заповнюються відповідні поля індексного файлу [296]. У найпростішому випадку ключовими словами служать назва та ім'я автора документа.

Серйозні обмеження при використанні цих систем пов'язані з тим, що пошук по ключових словах – це чіткий пошук, тобто користувач має вводити слова без помилок й не використовувати інші терміни з тотожнім змістом.

Поряд із традиційним індексуванням IP у даний час широко застосовуються *системи автоматичного реферування* [359]. Обсяг анотації має складати від 5 до 30% вихідного тексту (підготовка анотацій кількох джерел інформації вимагає ще більшого коефіцієнта стиску). Існують два типи таких систем. Головне розходження між ними полягає в тому, що вони формують – коротку анотацію чи набір цитат.

Текст, отриманий шляхом з'єднання окремих оригінальних фрагментів, позбавлений гладкості, його важко читати. Крім того, такі системи призначені тільки для обробки текстової інформації і не можуть працювати одночасно з кількома її джерелами. До подібних систем відносяться AutoSummarise (Microsoft Office) [138], Oracle Context, Insight Summarizer (AltaVista).

Формування короткого змісту тексту вимагає могутніх обчислювальних ресурсів для семантичної обробки природномовної інформації. Для цього звичайно застосовують один із двох основних підходів. Перший спирається на традиційний лінгвістичний метод синтаксичного розбору речень, а семантична інформація застосовується для анотування дерев розбору. Другий підхід ґрунтується на системах штучного інтелекту і спирається на розуміння природної мови (синтаксичний розбір є складовою частиною цього методу).

Зараз розробляються найрізноманітніші методи та відповідне ПЗ, призначені для синтаксичного та семантичного аналізу природномовних текстів з різних ПрО. Зокрема, програмний

комплекс КОНТЕНТ [282] призначений для здобуття з текстів основних фактів та подій на основі концептуального аналізу семантичних фреймів. Знання ПрО описуються як множина фреймів. Лінгвістичні знання подаються у вигляді словника. Визначення слова складається з його значення та синтаксичної інформації про це слово. Аналіз полягає у співвідношенні тексту та заданої ПрО.

Деякі програмні засоби здатні автоматично анотувати повнотекстові документи, з якими працює користувач, щоб використовувати цю інформацію для визначення контексту пошуку. Наприклад, система Watson моделює контекст інформації, що потрібна користувачеві, на основі вмісту документів, які він редагує засобами Microsoft Word чи переглядає у Internet Explorer. Ці документи аналізуються за допомогою евристичного алгоритму, який виявляє слова, характерні для вмісту документів. Потім знайдені слова автоматично додаються до запиту користувача. Крім того, Watson у фоновому режимі шукає в Web документи, пов'язані з матеріалами, що редагує та переглядає користувач. Недоліком системи є непрозорість алгоритмів, використовуваних системою, для кінцевого користувача.

Remembrance Agent [199] індексує певні файли (повідомлення електронної пошти, наукові статті тощо) та, поки користувач працює з якимось документом, веде пошук документів, пов'язаних з цим документом. До аналогічних рішень можна віднести Autonomy’s Kenjin [14], агентів Fab [15], Letizia [124] і WebWatcher [219], що вивчають область інтересів користувача для того, щоб запропонувати йому відповідні Web-сторінки. Спільним недоліком усіх цих систем є те, що вони не використовують стандартизовані засоби для опису ПрО, яка цікавить користувача, внаслідок чого знання, здобуті однією системою, не можна застосовувати в іншій.

Альтернатива індексації природномовної інформації – технологія, розроблена компанією Excalibur Technologies, яка поєднує метод адаптивного розпізнавання образів APRP (Adaptive Pattern Recognition Processing) і семантичні мережі. Вона дозволять працювати з інформацією будь-якого типу: текстом, графікою, відео тощо. Метод APRP використовує теорію нейронних мереж і дозволяє здійснювати бінарну індексацію, за якою розмір індексу навіть при обробці неструктурованої інформації не перевищує 30% від розміру вихідних даних [297]. Проте ця технологія потребує значних обчислювальних ресурсів.

Метаописи інформаційних ресурсів

Інший спосіб відображення відомостей про ІР пов’язаний з використанням їх метаописів, які подаються у структурованій формі, придатній для машинного оброблення, – певного фіксованого набору обов'язкових параметрів. Метаопис ІР має бути гнучким, тобто його при необхідності можна розширювати додатковими параметрами і зв'язками (але це може призвести до неоднозначного трактування цих параметрів та їх значень).

Певна метаінформація про кожен html-документ міститься в його метатегах (приміром, ключові слова, короткий опис документа, відомості про автора, дату створення, інструкції для пошукових роботів). Але такий метаопис – негнучкий та обмежений за можливостями. Через це він не дозволяє глибоко описувати контент документа і тому не придатний для ґрунтовного аналізу ІР.

Аналіз публікацій показує, що найперспективніші розробки стандартів подання метаінформації пов’язані з проектом *Semantic Web* консорціуму W3C [18, 233]. Сьогодні значну увагу привертає система опису ІР *RDF* (Resource Description Framework) [192], прийнята як стандарт у 1999 році консорціумом W3C.

RDF підтримують провідні виробники ПЗ і постачальники контента. На жаль, RDF-описи ще недостатньо поширені і для значної частини ІР відсутні. Призначення RDF – стандартизувати визначення і використання метаданих, що описують ІР Web, а також подання даних. RDF складається з засобів опису ІР і завдання схем RDFS [191]. RDF дозволяє описувати структуру сайту та пов'язану з ним ПрО. RDF описує ресурси у вигляді орієнтованого розміченого графа, тобто кожний ІР може мати властивості, які у свою чергу також можуть бути ресурсами або їх колекціями.

RDF використовує базову модель даних «*об'єкт – атрибут – значення*». Цей базовий блок можна подати як ребро графа з міткою А, яке з'єднує вузли О та V, та позначити як $[\mathrm{O}] \xrightarrow{A} [\mathrm{V}]$. Дана нотація дозволяє змінювати місцями об'єкти і значення. Таким чином, будь-який об'єкт може відігравати роль значення.

Крім того, об'єкти та значення трійок RDF можуть також бути трійками, тобто графи можуть бути вкладеними. Завдяки цьому RDF може оперувати твердженнями. Це дозволяє, наприклад, відображати сумнів або згоду з твердженнями, створеними іншими користувачами, а також вказувати, що даний об'єкт має певний тип. Вираз «ІР R_1 як властивість Р має ІР R_2» можна проінтерпретувати як

предикат $P(R_1, R_2)$, а потім використовувати це твердження як об'єкт для інших тверджень. Така інтерпретація дозволяє описувати за допомогою RDF концептуальну інформацію. Таким чином, RDF здатний виконувати роль універсальної мови опису семантики ресурсів і взаємозв'язків між ними. Стандарт RDF використовує XML-синтаксис. Він складається з двох основних засобів:

- опису IP, які визначають модель об'єкта, та зв'язків між ними;
- схем RDFS, за якими описується IP.

Метадані можуть бути:

- вбудованими в сам IP;
- зберігатися незалежно від IP.

Реалізуючи перший підхід, багато виробників ПЗ випускають продукти, які автоматично формують деякий невеликий блок RDF-опису всередині документа (приміром, редактор Microsoft Word 2000). Другий підхід більш універсальний, тому що дозволяє додавати метадані до будь-яких ресурсів. Практично реалізовувати цей підхід почали в 2002 р. на базі одного з найбільших каталогів Інтернету Open Directory [167] в рамках проекту автоматичного створення репозиторію RDF-описів IP Інтернету.

Метадані, які розміщуються окремо від ресурсу, краще зберігати та передавати у форматі XML, максимально використовуючи можливості моделі RDF для забезпечення інтероперабельності. Обмін метаданими зводиться до пересилання RDF/XML-файлів.

RDF Schema визначається в термінах базової інформаційної моделі RDF – структури графа для опису ресурсів і властивостей. Всі словники RDF використовують базову структуру, що описує класи ресурсів і типи зв'язків між ними. Це дозволяє застосовувати різнорідні децентралізовані словники, створені для машинної обробки за різними принципами.

RDF Schema дозволяє розробникам визначати конкретний словник для даних RDF (такий, як authorOf) і вказувати види об'єктів, до яких можуть застосовуватися ці атрибути, тобто механізм RDF Schema надає базову систему типів для моделей RDF, яка використовує такі терміни, як Class, subPropertyOf і subClassOf, для схеми, яка орієнтована на конкретне застосування. Вирази схеми RDF – коректні вирази RDF.

Об'єкти RDF можна визначити як екземпляри одного або кількох класів за допомогою властивості type. Властивість subClassOf дозволяє розробникам вказувати ієрархічну організацію таких класів, а subPropertyOf виконує те ж саме для властивостей. Обмеження на

властивості також можуть бути зазначені за допомогою конструкцій домена і діапазону, що застосовуються як для розширення словника, так і для інтерпретації виразів RDF.

На основі RDF Schema створено RDF Vocabulary Description Language 1.0: RDF Schema, де RDF використовується для опису RDF-словників. Мова опису словника RDF визначає класи і властивості, які використовуються для опису інших класів і властивостей.

Важливою особливістю стандарту RDF є розширюваність: на RDF можна задати структуру опису джерела, використовуючи і розширюючи вбудовані поняття RDF-схем, такі, як класи, властивості, типи, колекції. Модель схеми RDF включає спадкування класів і властивостей.

Dublin Core Metadata Elements

Щоб спростити та уніфікувати створення метаописів ІР, користувачам потрібні певні шаблони та стандарти опису типових ІР. Інакше один автор напише, приміром, "Назва", інший – "Заголовок", а третій – "Title". Сьогодні найбільш розповсюджений набір елементів для створення метаданих, розроблений міжнародною групою "*Dublin Core Metadata Elements*". Він складається з 15 елементів [52, 53], які можна розбити на три групи елементів:

- *Content* – відносяться до змісту ресурсу:
  - заголовок – Title;
  - предмет – Subject;
  - опис – Description;
  - тип – Type;
  - джерело – Source;
  - відношення – Relation;
  - сфера дії – Coverage;
- *Intellectual Property* – характеризують інтелектуальну власність:
  - розробник – Creator;
  - видавець – Publisher;
  - співробітник – Contributor;
  - права – Rights;
- *Instantiation* – описують конкретний екземпляр ресурсу:
  - дата – Date;
  - формат – Format;
  - ідентифікатор – Identifier;
  - мова – Language.

Цей набір елементів можна розширювати з використанням наявних стандартів. Деякі елементи основного комплекту опису потребують більш детального розкриття для запобігання неоднозначної інтерпретації. Щоб збільшити деталізацію і складність описів, зберігаючи сумісність з "Dublin Core Metadata Elements", різні організації розробляють розширення та додаткові кваліфікатори для деяких базових елементів.

Розглянемо детальніше призначення деяких елементів.

Елемент *Subject* визначається за допомогою двох тезаурусів: предметного та/або функціонального. *Предметний тезаурус* містить поняття ПрО і відображає зміст документа, він є відповіддю на питання "Про що цей документ?". *Функціональний тезаурус* відображає роль документа в людській діяльності і відповідає на запитання "Для чого цей документ?".

Елемент *Type* відображає жанр ресурсу та категорію його змісту. Можна обрати одну зі стандартних категорій: text, image, sound, dataset, software, interactive, event, physical object. Цей список можна розширити (кожен елемент поділяється над піделементи), наприклад, конкретизувати *event* як конференцію, семінар, виставку.

Елемент *Format* відображає середовище, формат даних IP, матеріал, з якого складається IP (якщо це фізичний об'єкт), і, можливо, його фізичні розміри. Якщо ресурс подано в електронному вигляді, тоді його формат рекомендується вибирати зі списку стандарту MIME. Приклади електронних форматів: text/xml – текст у форматі XML; text/plain – текст без форматування; image/gif – зображення у форматі GIF. Для **інших** IP формат рекомендується вибирати зі списку фізичних об'єктів, наприклад з AAT (Art and Architecture Tesaurus).

На сьогодні розроблено ряд програмних продуктів, які дозволяють створювати RDF-описи різних джерел (наприклад, RDFPic додає RDF-описи до зображень), здійснювати валідацію RDF-описів документів (RDF-validator) та редагування (RDF-editor) [189]. Передбачається інтеграція концепції RDF-бази з форматами TopicMaps, XML Topic Maps (XTM), RSS, MPEG.

Для того щоб задавати запити до RDF-джерел даних, розроблені спеціальні мови – приміром, RDF Query [190], яка використовується в проекті Sesame [207].

Зараз існує досить вільно розповсюджуваний простий та зручний програмний засіб для створення та редагування RDF-описів

IP, який підтримує набір елементів "Dublin Core Metadata Elements", – RDFEdit (рис.3.10).

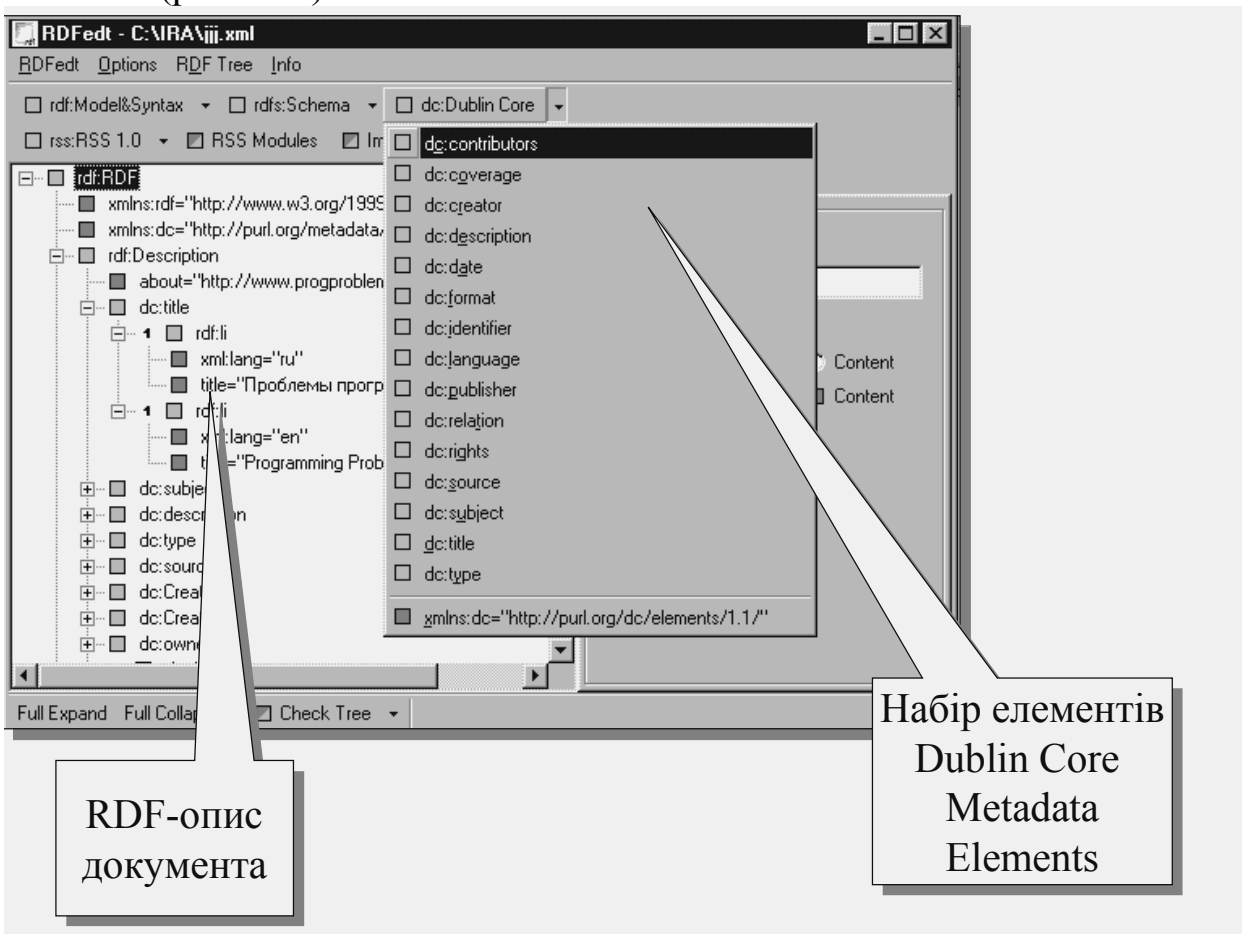


Рис.3.10. Введення значень елементів Dublin Core за допомогою RDFEdit

Цей програмний засіб підтримує введення елементів Dublin Core, забезпечує перевірку правильності метаопису документа, який можна зберегти у форматі XML. Приміром, RDFEdit використовувався для створення коректного метаопису сайту електронної версії журналу "Проблеми програмування" (www.pprogproblems.org.ua). БД, в якій містяться відомості про публікації, реалізовано на MySQL [357]. Доступ до інформації в БД здійснюється через локальну пошукову машину, реалізовану мовою PHP [352] (рис.3.11). Наявність RDF-опису забезпечує сайту перше місце в результатах пошуку за ключовими словами "проблеми програмування" (слід помітити, досить поширеними серед IP Інтернет) у таких ІПС, як Rambler та AltaVista.

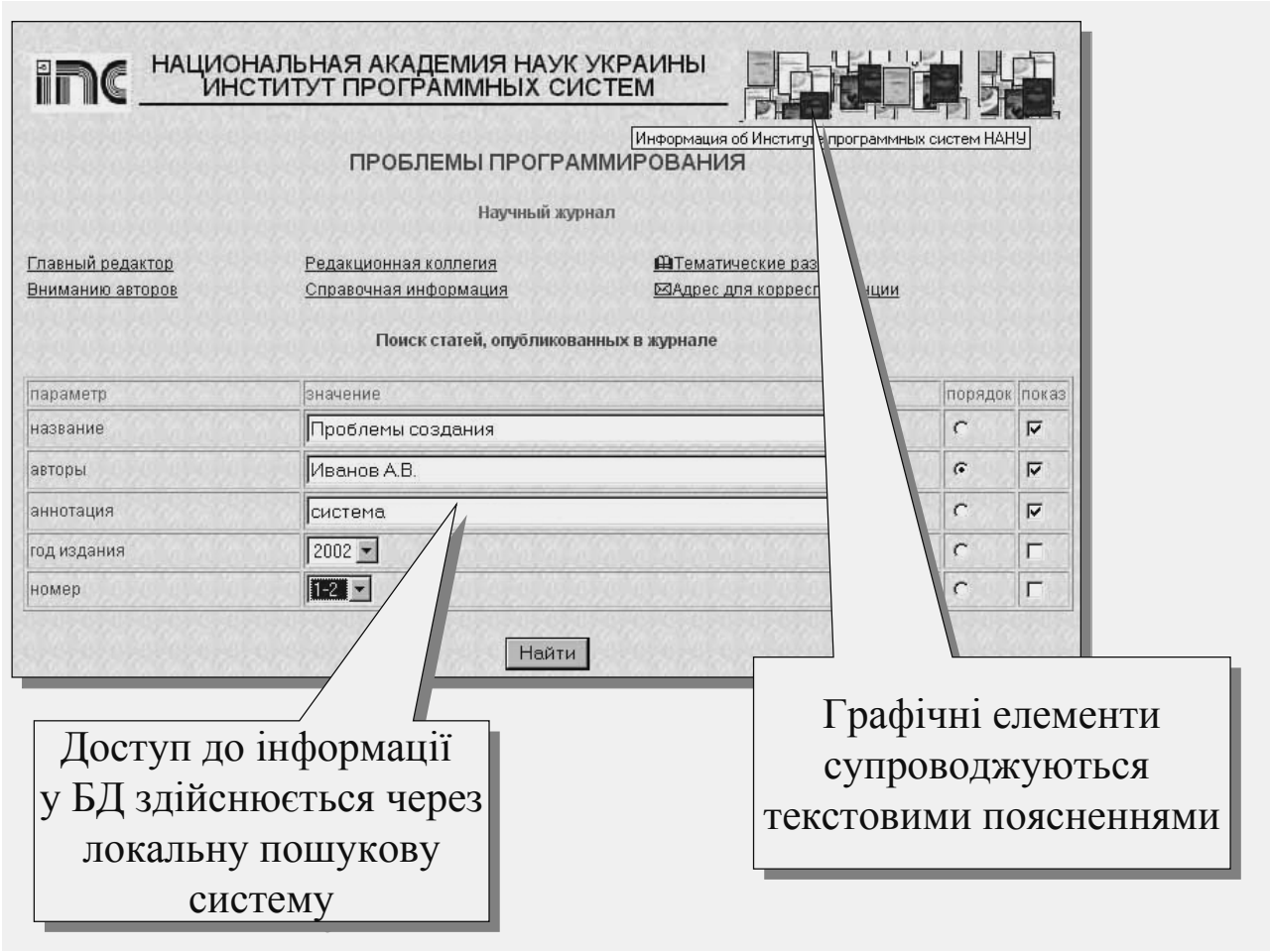


Рис.3.11. Електронна версія журналу "Проблеми програмування"

Проект Semantic Web

Створення універсальних засобів семантичної обробки інформації шляхом інтеграції усіх наявних підходів є метою проекту *Semantic Web* [233] консорціуму W3C. Результати роботи об`єднаної групи Semantic Web використовуються для автоматичного створення RDF-описів і сховищ метаданих на базі Open Directory [167] пошуковим механізмом Google та у новому проекті консорціуму W3C SWAD-Europe [214], що займається проблемою зв'язку сховищ семантичних даних з існуючими реляційними БД. Велика увага в цьому проекті приділяється моделюванню розподіленого середовища Інтернету та архітектурі метаданих [136, 25]..

В основу Semantic Web покладене абстрактне подання IP Інтернету на базі RDF з урахуванням інших поширених стандартів: специфікацій MCF (Meta Content Framework), PICS (Platform for Internet Content Selection), XMI (Metadata Interchange Format), P3P (Platform for Privacy Preferences) тощо.

У рамках проекту Semantic Web задіяні передові інформаційні технології: агентно-орієнтований підхід у програмуванні – проект DAML+OIL (DARPA Agent Markup Language + The Ontology Inference Layer) [1], онтологічні системи [234], XML [48] тощо. Консорціумом W3C розроблено ряд низькорівневих протоколів роботи Web-агентів, а також мова опису сервісів агентів WSDL (Web Service Description Language) [237], протокол обміну інформацією між програмними агентами SOAP (Simple Object Access Protocol) та специфікація UDDI (Universal Description, Discovery and Integration), що пропонує користувачам уніфікований і систематизований засіб пошуку постачальників послуг через централізований реєстр Web-служб.

Для того щоб формально відобразити в метаописі IP знання ПрО, до якої відноситься цей ресурс, у проекті Semantic Web використовують онтологічні системи, що містять ієрархію концепцій ПрО й описують важливі властивості кожної концепції за допомогою механізму «атрибут – значення». У Semantic Web машинна обробка змісту IP здійснюється шляхом розмітки документів за допомогою онтологічних термінів. Онтології дозволяють концептуалізувати домен фіксуванням сутностей і зв'язків між ними. Для формального подання онтологій розроблено мови онтологій DAML+OIL та OWL. Обидві мови засновані на RDF і RDF Schema.

DAML+OIL – це семантична мова розмітки Web-ресурсів, яка розширює стандарти RDF і RDF Schema більш повними примітивами моделювання. Онтологія DAML+OIL – колекція RDF-трійок. Мова DAML+OIL забезпечує багатий набір конструкцій для створення онтологій і розмітки інформації. Онтологія DAML+OIL містить:

- заголовки (headers);
- елементи класів (class elements);
- елементи властивостей (property elements);
- екземпляри (instances).

Класифікація онтологій DAML *Ontology Library* складається з груп онтологій, об'єднаних за наступними параметрами:

- унікальним ідентифікатором ресурсу URI;
- датою подання;
- ключовими словами;
- каталогом Open Directory Category;
- класами;
- властивостями;
- використовуваними просторами імен;
- джерелом фінансування (організацією);

- підлеглими організаціями.

В OIL онтології містять визначення властивостей (slot-def) і класів (class-def). Slot-def описують відношення між об'єктами.

Для рішення задачі семантичної інтероперабельності Web у складі проекту Semantic Web використовується мова подання онтологій OWL (докладніше у главі 1), що походить від DAML+OIL і також базується на RDF. Мова подання онтологій OWL розширює можливості XML, RDF, RDF Schema та DAML+OIL. Цей проект знаходиться на стадії розробки, але його розробники передбачають створення потужного механізму семантичного аналізу. Основні переваги OWL порівняно з DAML+OIL полягають в усуненні деяких обмежень та тих конструкцій DAML+OIL, які не використовуються, а також у здатності прямо вказувати симетричність властивості.

3.3. Інформаційно-пошукові агенти

Інформаційно-пошукові агенти (ІПА) – це агенти, метою функціонування яких є ефективна взаємодія користувача з інформаційним середовищем і перетворення останнього в персоналізовані знання для конкретних користувачів.

Використання ІПА забезпечує користувачам значні переваги порівняно з такими традиційними засобами пошуку, як ПМ:

- користувач отримує тільки найбільш релевантні результати пошуку, попередньо переглянуті ІПА з урахуванням специфіки ПрО, яка цікавить користувача, та його персональних уподобань;
- результати пошуку зберігаються для подальшої обробки та аналізу;
- можна автономно (наприклад, за розкладом) виконувати постійні інформаційні запити користувача без явних його вказівок;
- ІПА може коригувати свою поведінку за власним досвідом;
- співпраця ІПА підвищує ефективність виконання запитів;
- в процесі пошуку використовуються знання ПрО, подані як словники, тезауруси та онтології, а також методи дедуктивного, індуктивного та традуктивного виведення.

Всі ці дії виконуються автоматично, для обробки посилань може запускатися відразу кілька процесів. Це забезпечує оптимальне використання каналу Інтернету і значне зниження витрат часу.

Таким чином, можна розглядати ІПА як інтелектуальну надбудову над ПМ. Такі агенти – досить складні системи, тому користувачеві важко прогнозувати їх поведінку та розуміти шляхи отримання результатів. Це призводить як до технічних, так і до

соціальних проблем, вирішити які можна через створення адекватної теоретичної моделі .

ІПА мають забезпечувати:

- пошук інформації згідно запиту користувача (одноразовому та постійному);
- можливість автономного виконання завдань користувача;
- можливість доставки інформації користувачеві;
- фільтрацію постійних потоків інформації;
- надання інформації, яка може зацікавити користувача щодо заданого запиту.

Агент працює постійно, виконуючи водночас декілька завдань користувача. Модель ІПА містить (рис.3.12) [299]:

- модель користувача, яка формується та змінюється в процесі взаємодії агента з користувачем;
- моделі ІР (постачальників інформації);
- динамічну модель ПрО, яка цікавить користувача (онтологію).

БЗ ІПА містить таку інформацію:

- модель користувача;
- модель навколишнього середовища, в якому діє ІПА;
- моделі ІР, до яких ІПА має доступ;
- інформацію про доступні засоби впливу на навколишнє середовище;
- інформацію про інші ПА, з якими можна взаємодіяти для виконання своїх цілей,
- модель ПрО, що цікавить користувача.

Планувальник забезпечує планування та здійснення пошуку, автоматичне поповнення БЗ та взаємодію з іншими агентами.

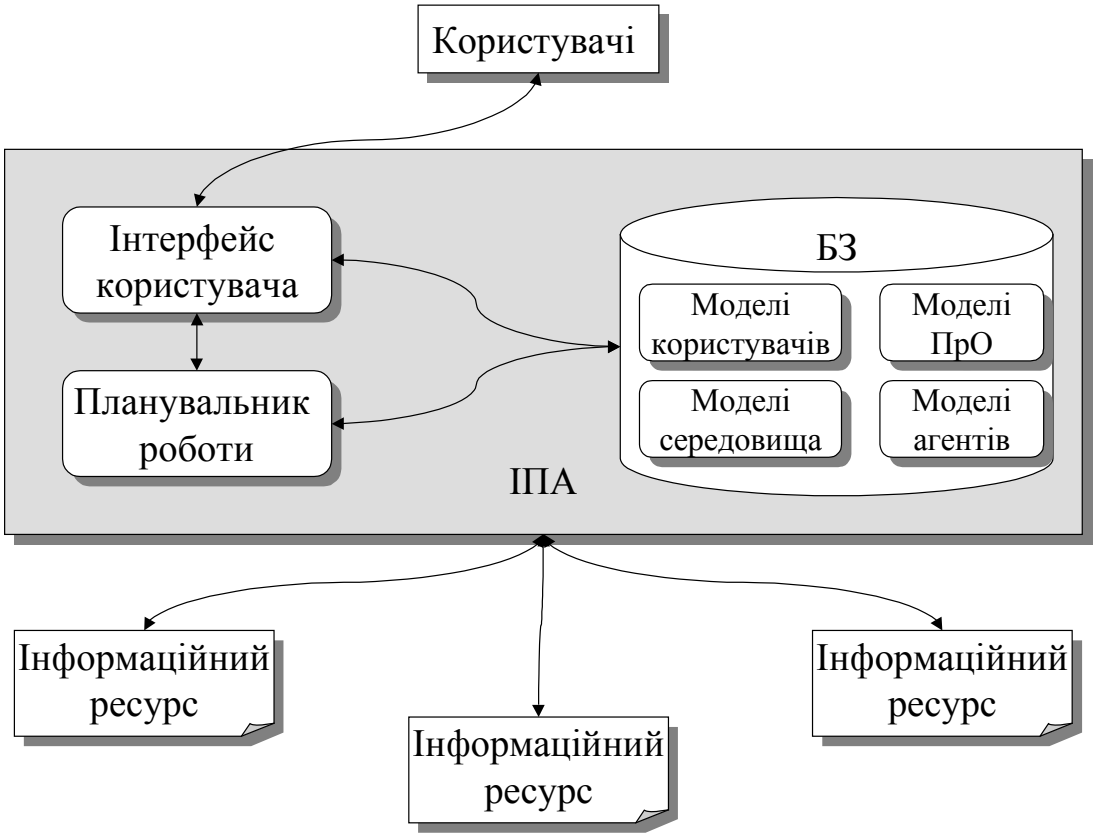


Рис.3.12. Структура ІПА

Інтерфейс користувача потрібний для отримання завдань від користувача і повернення йому результатів їх виконання. Крім того, він може використовуватися для поповнення моделі навколишнього середовища.

ІПА потребує знань про модель інформаційних потреб користувача з метою:

- ефективнішого пошуку;
- проактивності – пошуку інформації, яка за результатами аналізу постійних запитів може зацікавити користувача;
- спостереження за потоками інформації.

Модель відображає постійні інформаційні потреби користувача (інформаційні запити про зміни у вказаних джерелах, постійні запити без зазначення конкретних джерел, історію відповідей на постійні запити, розклад запитів тощо). Вона також має містити інформацію про те, яким інформаційним джерелам віддає перевагу користувач (приміром, користувача цікавить інформація тільки на сайтах певного міста або тільки українською мовою). Наявність в агента знань ПрО підвищує ефективність пошуку.

Модель користувача

Інформація про користувача містить такі елементи:

- постійні користувацькі запити;
- теми, що цікавлять користувача;
- джерела інформації;
- історію одноразових користувацьких запитів;
- специфікація документів, яким надає перевагу користувач (якщо треба, то для кожного з постійних запитів окремо);
- заборонені джерела.

Всі ці дії виконуються автоматично, для обробки посилань може запускатися відразу кілька процесів. Це забезпечує оптимальне використання каналу Інтернету і значне зниження витрат часу.

Модель інформаційного середовища ІПА

ІПА здатний враховувати довгострокову зацікавленість користувача в інформації з однієї або кількох вузьких ПрО і специфіку цих областей, а також переваги конкретного користувача.

Модель інформаційного середовища ІПА містить [299]:

- ІР, їх властивості, тематика та правила роботи з ними;
- засоби отримання інформації про навколишнє середовище;
- засоби впливу на навколишнє середовище.

Поповнення моделі інформаційного середовища агента може здійснюватися безпосередньо користувачем або автоматично у процесі роботи.

Відомості про інформаційне середовище можна отримати в результаті виконання процесу пошуку [285]. Моделлю інформації, яку намагається одержати користувач, є запит.

Поширені типи пошуку:

- пошук документа, інформація про який вже міститься в БД агента (приміром, ІПА має перевірити, чи змінилася html-сторінка, яка відповідає певній URL-адресі);
- пошук документа у рамках певної підобласті Інтернету (приміром, ІПА має здійснити перевірку усіх гіперпосилань, що містяться у певному html-документі або групі документів, пошук в електронній бібліотеці, пошук на сайті, що має локальну ПМ);
- пошук документів в Інтернеті (за допомогою ПМ, каталогів тощо);
- пошук нових сайтів та інших ІР.

Модель інформаційного середовища ІПА містить засоби впливу на навколишнє середовище, але ці засоби не є принциповими для вирішення проблеми пошуку і мають тільки допоміжне значення.

Приміром, відвідання користувачем якогось сайту, посилання на який містилось у результатах пошуку, фіксується лічильником й впливає на статистичні дані, що використовуються для подальших модифікацій цього сайту. Інший приклад – реєстрація користувача на знайденому сайті та заповнення різноманітних анкет. Ця інформація теж може змінювати інформаційне середовище, але досить слабо.

Приклади застосування агентних технологій для пошуку інформації

Зараз багато розробників ПЗ звертаються до використання агентно-орієнтованих технологій для створення інформаційно-пошукових механізмів. На сьогодні розроблено багато ІПА, більшість з яких – безкоштовні. Вони різняться за рівнем інтелектуальності, ІПМ, інтерфейсом тощо. Таке різноманіття вказує на перспективність цього підходу та актуальність розробки його теоретичних засад. Розглянемо кілька відомих мультиагентних ІПС (на основі інформації, що наведена у [265, 293, 332] та відомостей, які містяться на офіційних Інтернет-сайтах відповідних компаній). Через те, що певна частина розглянутого ПЗ – комерційна, її опис базується на характеристиках, наданих розробниками.

Copernic (www.copernic.com), працює з багатьма ІПС, має зручний інтерфейс користувача і дозволяє перевіряти доступність знайдених посилань [41]. Посилання, що дублюються, відкидаються, а інші упорядковуються за рівнем релевантності. Недоліки програми – неможливість відключення реклами і те, що інформація про нові ІПС може додаватися тільки розроблювачем.

Copernic використовує досить простий механізм пошуку: отримані від різних ПМ результати поєднуються і сортируються . При цьому інформація про знайдені посилання одержується безпосередньо від ПМ, і невідомо, чи є вона актуальною і релевантною. Як критерій пошуку система сприймає тільки прості запити (без логічних операторів), однак при подальшій обробці знайдених посилань можна модифікувати запит із уведенням булевих виразів.

Більш вдалою в цьому відношенні є програма *SSSpider* (Subject Search Spider), яку її розробники відносять до ІПА другого покоління: результати, отримані від пошукових машин, відразу перевіряють на доступність. Некоректні і дубльовані посилання знищуються, після цього програма перевіряє наявність на знайдених сторінках ключових слів і готує звіт на основі актуальної інформації.

SSSpider дозволяє використовувати багато мов для створення запитів, у тому числі і російську. Програма дозволяє користувачу самостійно підключати пошукові служби.

Ця програма, також як і Copernic, може обробляти тільки прості запити (можна шукати всі ключові слова, будь-яке з них або фразу цілком). Проте у багатьох випадках зручніше використовувати формальні запити, створені за допомогою логічних операторів). Проблема полягає в тому, що синтаксис таких запитів не стандартизований і у різних ПМ використовуються різні позначення (приміром, логічне "і" позначають як "&", "^", "+" або "AND").

Mata Hari (www.thewebtools. com) забезпечує роботу з формальними запитами. Запит, розмічений за допомогою логічних операторів, дужок і лапок, може бути переданий до багатьох ПМ. Якщо якась з них не розуміє формальних запитів, їй направляється звичайний запит, а логічна обробка здійснюється локально. Mata Hari працює аналогічно SSSpider: автоматично завантажує знайдені сторінки і перевіряє їх актуальність та релевантність. Головний недолік програми – нерозуміння української та російської мов.

Пошук з використанням усіх припустимих ПМ займає багато часу, тому можна вказувати максимальну кількість посилань, що повертаються кожною ПМ, мінімальний і максимальний розмір документів, дату останньої зміни тощо. Можна використовувати певну підмножину ПМ. Для цього виділені спеціальні групи ПМ, у які ІПС об'єднані за тематикою (Computers, Business-Finances, Games, Graphics тощо) або за деякими іншими принципами (існує, наприклад, група Starting Point, до якої входять найбільш популярні ПМ, та група Metasearchers, яка поєднує метапошукові системи). Користувачам дозволяється створювати власні групи.

Ще одна корисна властивість цієї системи – якщо на знайденій сторінці є гіперпосилання, що можуть зацікавити користувача, то вони також будуть перевірені на відповідність запиту.

*Kenjin***,** на відміну від інших інтелектуальних ІПА, працює не з ключовими словами, а з цілим текстом.

Tryllian (www.tryllian.com) – компанія, яка спеціалізується на виробництві Інтернет-застосунків. Ця компанія розробила систему для створення MAC Tryllian Agent Developer Kit (http://www.tryllian. com/development/index.html).

Cybion (www.cybion.com) – французька IT-компанія, яка займається розробкою і впровадженням інтелектуальних програмних продуктів для пошуку, аналізу і менеджменту інформації. Її

продукція – адаптовані під потреби конкретного користувача інтелектуальні сервіси Cybion Eye, Cybion Report тощо.

ТОВ "*НейрОК*" (www.neurok.ru) – розроблювач інтелектуальних систем обробки інформації на основі нейронних мереж, у тому числі ПА, які шукають інформацію в Інтернеті й аналізують результати пошуку. Розробки базуються на платформі *NeurOK Semantic Suite* [156] – наборі серверів застосувань і програмних модулів, що вирішують найпоширеніші задачі по доставці, сортуванню, збереженню і пошуку текстової інформації. Діалогова пошукова система за ключовими словами уточнює контекст запиту. Система сама пропонує варіанти уточнюючих термінів, роблячи пошук більш спрямованим. Для організації діалогу з користувачем використовуються семантичні технології НейрОК. Однією з функцій серверного компонента NeurOK Semantic Engine є побудова за будь-яким фрагментом тексту набору асоціативно пов'язаних з ним слів.

BinNet (www.binnetcorp.com) спеціалізується на виробництві Java- і Prolog-застосувань, а також інструментарію для виробництва інтелектуальних мобільних програмних продуктів для Інтернету (Intelligent Mobile Agent Programming ToolKit).

Науково-виробнича компанія "*Генезис знань*" (www.kg.ru) – одна з провідних російських компаній в галузі досліджень і розробок віртуальних світів і інтелектуальних агентів. На сайті можна одержати докладну інформацію про діяльність фірми і програмні продукти, а також отримати консультації.

Мультиагентна ІПС *Autonomy* [14] забезпечує користувачів інтелектуальними засобами пошуку інформації, релевантної його інтересам. Система дозволяє впорядковувати знайдені в Інтернеті документи за темами і автоматизувати сам процес пошуку.

МАС Autonomy – набір ПА для інтелектуального пошуку й обробки інформації, що організовані в рамках спеціалізованої програмної оболонки. Система більше орієнтована на кінцевих користувачів, а не на спеціалістів у певній галузі. Інтерфейс системи Autonomy досить простий та інтуїтивно зрозумілий.

Запит на пошук інформації в системі Autonomy задається у вигляді природномовного тексту. Система автоматично аналізує цей текст і здобуває з нього зміст. У цій ІПС використовується технологія нейронних мереж, а також метод подання ПрО із застосуванням алгоритмів розпізнавання образів і обробки сигналів. При цьому системою формується образ документа, що є релевантним запиту, який використовується на наступних етапах пошуку.

Пошук здійснюється за допомогою методів нечіткої логіки. В основі пошукового алгоритму лежить механізм динамічних міркувань. Важливою особливістю Autonomy є наявність режиму навчання пошукових агентів.

Intelliseek (www.intelliseek.com) – провідний виробник інтелектуальних on-line пошукових систем (Intelligent Applications™), орієнтованих на конкретного користувача, які забезпечують фільтрацію інформації, текстовий аналіз, навчання ПА, тематичну автоматичну категоризацію. Найвідоміша розробка цієї компанії – пошукова система BullsEye®, яка працює на стороні клієнта.

Інтеграція агентних технологій до набору технологій пошуку та трекінгу iAsk™, що забезпечують знаходження відповідей на питання природною мовою, призвело до створення потужного пошукового інструментарію, який забезпечує кінцевому користувачу можливості задавати спеціалізовані питання та отримувати спеціалізовані відповіді замість набору можливих відповідей. Це дозволяє обробляти не тільки ліцензований фінансовий контент та ринкові повідомлення, а також релевантні зв'язки з відповідними Web-сайтами.

Інтелектуальний пошуковий агент *Personal Finder* здійснює пошук потрібної користувачу інформації як у Інтернет/Інтранет, так і на локальному комп'ютері користувача. Під час пошуку агент здійснює опитування спеціалізованих ПМ, що працюють у відповідній ПрО.

Слід вказати ще кілька відомих досить простих метапошукових ПА. Кожний з них працює з кількома ІПС і видає загальний покажчик знайдених посилань. *WebCompass* [235] і *WebSeeker* [236] дозволяють складати розклад для періодичного виконання та відновлення пошуку.

EchoSearch [54] корпорації Iconovex працює швидше за рахунок малої кількості ПМ, до яких він звертається. *WebSeeker* фірми ForeFront Group дозволяє виконувати ітеративний пошук.

Пошукова система *Webcompass* [235] забезпечує інтегровані засоби автоматизованого пошуку релевантної користувальницькому запиту інформації серед Інтернет-ресурсів, але проектні рішення, які обрали її розробника, істотно відрізняються. Це пов'язано з орієнтацією на іншу категорію користувачів, – фахівців у певній ПрО, які здатні формувати структурний опис області своїх інтересів.

Webcompass складається з агентно-орієнтованих компонентів, які підтримують всі основні процеси пошуку й аналізу інформації. Опис ПрО в системі Webcompass базується на використанні

таксономії понять, пов'язаних між собою такими відношеннями, як "is a", "part of", "has part", "is a kind of" тощо. Важливе обмеження системи: між двома поняттями не може бути більше одного відношення.

Запит до системи Webcompass базується на безпосередньому використанні побудованого користувачем опису ПрО. Цей опис подано як таксономію понять – ключові слова і вирази, тому для формування запиту користувачеві досить відмітити потрібні теми. За цими позначками система автоматично формує запит.

Пошук у системі Webcompass ведеться за ключовими словами відразу на кількох десятках ПМ. Користувач може змінювати список джерел, додавати адреси для пошуку в Інтранеті, Usenet, FTP тощо. Система автоматично перевіряє знайдені посилання і складає резюме документів, а також визначає ступінь їх відповідності запиту.

WebMachine (www.webmachine.ru) – інтелектуальний метапошуковий ПА, який одночасно здійснює пошук на кількох серверах і вміє зберігати й обробляти результати пошуку. Процес пошуку максимально спрощений і інтуїтивно зрозумілий. WebMachine містить спеціальний механізм, що дозволяє правильно сформувати запит. Програма підтримує інтерфейс російською мовою.

Розробка TransCom Software – пошуковий агент *BeeLine* (www.beeline4oz.com) відрізняється простотою експлуатації і зручною системою настроювань. BeeLine підтримує інтерфейс п'ятьма мовами, у програму убудовані автоматична перевірка орфографії і словник синонімів. Після обробки запиту знайдені посилання автоматично перевіряються на доступність і ті, що не існують або дублюються, видаляються.

BeeLine дозволяє виконувати тільки прості запити, однак функція Exclude дозволяє вказувати слова, які повинні бути відсутніми у результатах пошуку.

Інтелектуальний ІПА *DigOut4U* (www.arisem.com/en/index.html) французької компанії Arisem підтримує природну мову (англійську і французьку). Характерна риса технології пошуку запитуваних документів – семантичний аналіз.

Інтелектуальний ІПА *DirectSeek* (www.directseek.net) одночасно опитує сотні ПМ і надає користувачеві добірку матеріалів з високим ступенем релевантності. Тематично орієнтовані ПА, реалізовані в DirectSeek, забезпечують глибокий пошук інформації з тематики запиту користувача, досліджуючи сайти, конференції, форуми, архіви новин, on-line аукціони і БД.

CyberAlert (www.cyberalert.com/overview.html) – цілком автоматичний Інтернет-сервіс, що знаходить потрібну інформацію та регулярно відслідковує більш 3500 web-публікацій, 63000 конференцій і форумів. Текстовий аналіз і потрійна фільтрація дозволяють досягти високої релевантності результатів пошуку.

MySimon (www.mysimon.com) – інтелектуальний ПА, побудований за технологією "The Virtual Learning Agent", призначений для пошуку в on-line магазинах. MySimon здійснює інтелектуальний пошук, порівнюючи ціни товарів у різних on-line магазинах. Агент має простий інтерфейс, здійснює швидкий і релевантний пошук.

MP3-Wolf (www.trellian.com/mwolf/index.html) – пошуковий робот, що сканує Інтернет у пошуках потрібних користувачу мультимедійних ресурсів (mp3, midi, wave і інших музичних файлів). Робот працює в режимі реального часу і здатний знаходити, сортувати й аналізувати десятки тисяч музичних файлів і посилань у годину, відкидаючи при цьому застарілі посилання. Він може досліджувати кілька сайтів одночасно.

ПА *Clickthebutton* (www.clickthebutton.com) призначений для полегшення покупок в режимі on-line. Коли користувач знаходить товар, який він бажає купити, Clickthebutton пропонує список інших сайтів, що продають даний продукт, указуючи також і ціну. Агент здатний порівнювати ціни на продукти у різних Інтернет-магазинах.

Система *MARRI* призначена для пошуку Web-сторінок, релевантних запитам у певній ПрО. У MARRI використовується онтологічне подання знань. Онтологія розуміється як множина концептів і зв'язків між ними.

MARRI містить кілька типів ПА: агенти-брокери, агенти мережі, інтерфейсні і спеціалізовані агенти. Агенти мережі використовують для попереднього добору інформації стандартні ПМ. Потім спеціалізовані агенти аналізують отримані ІР, а після цього порівнюють їх з онтологією ПрО. Користувачу надаються тільки ті ІР, що успішно пройшли цю онтологічну перевірку.

Кожний з ПА має наступні властивості:

- є автономною Java-програмою з власною мережною URL-адресою;
- взаємодіє з іншими агентами за допомогою мови ACL (Agent Communication Language);
- є споживачем і постачальником інформації залежно від того, з якими агентами системи він спілкується;

- може взаємодіяти з такими автономними програмними компонентами, як браузери, аналізатори природної мови та онтологічні БД.

Онтологічна перевірка тексту складається з кількох етапів. Спочатку здійснюється морфологічний і синтаксичний аналіз тексту. Потім будується синтаксичне дерево тексту. Після цього визначається тип речення та тип мовного акта, що відповідає цьому реченню. Використовуються тільки прості стверджувальні речення. Іменники порівнюють з концептами онтології, а дієслова – з ролями.

Агент інтерфейсу підтримує інтелектуальну взаємодію з користувачем. Він допомагає сформувати запит, а потім представляє результати пошуку як список адрес релевантних сторінок. Коли користувач вибирає цікавлячу його ПрО, цей агент запитує у БД відповідну онтологію і інформує інших агентів мережі про те, яка онтологія буде використовуватися. Задача агента мережі – перевірка, завантаження й аналіз Web-сторінок.

У MARRI використовуються два типи агентів-брокерів: URL і HTML. Брокери URL призначені для підтримки списків Інтернет-адрес, що поставляються браузером, а HTML – для розподілу Web-сторінок між агентами обробки тексту для подальшого аналізу.

Метою функціонування агентів обробки тексту є семантичний аналіз Web-сторінок для перевірки їхньої релевантності на основі відповідної онтології. Попередньо ці агенти перетворюють HTML-текст у структуроване подання, з яким можуть працювати морфологічний і синтаксичний аналізатори.

Система *OntoSeek* [166] розроблена для здобуття інформації з “жовтих сторінок” і каталогів у режимі on-line. Для точного опису IP використовується обмежена кількість термінів природної мови. Для подання запитів і опису ресурсів замість списків типу “атрибут-значення” використовуються прості концептуальні графи, для яких проблема контекстного ототожнення зводиться до керованого онтологією пошуку в графі. Якщо онтологія показує, що між вузлами і дугами існує задане відношення, то вони порівнянні. Система базується на лінгвістичній отології, тому вузли графа мають бути прив'язані до відповідних лексичних одиниць.

Проект *SAIRE* [204] – це заснований на агентах механізм інформаційного пошуку. Він забезпечує інтегрований доступ користувачів до розподілених джерел даних, у тому числі – до електронних бібліотек NASA і NOAA. SAIRE дозволяє задавати

запити природною мовою (мова системи – англійська) з різних розділів науки, використовуючи спеціальні терміни.

Amalthaea – МАС для фільтрації, виявлення і спостереження ІР. Взаємодія агентів здійснюється відповідно до ринкової економічної моделі. Amalthaea може вивчати інтереси конкретних користувачів і пристосовуватися до них.

Мультиагентна метапошукова система "*Метафора"* (http://metafora.nm.ru.) містить три класи незв'язаних агентів: агенти даних (D-агенти), пошукові агенти (S-агенти) і агенти новин (N-агенти), а також підсистему репозиторію (БД) і арбітр ресурсів. Основне місце в системі займають агенти даних. Кожному D-агентові відповідає певний документ. Кожен D-агент характеризується номером, двома зваженими векторами (генотипом ПА, тобто набором ключових термінів документа, та його фенотипом – набором посилань на S-агенти) і скалярною величиною – кількістю ресурсів. Він може знаходитися в трьох станах: сплячому, активному і збереженому пошуку.

У процесі пошуку можуть використовуватися різні ІР: ІПС, електронні бібліотеки тощо. Кожному з них відповідає S-агент, що має інформацію про її інтерфейс і властивості.

Агенти новин забезпечують інтерфейс між МАС і іншими джерелами інформації. БД містить відомості про ПА і документи: адреси, стан обробки, релевантність тощо.

Керуючий агент контролює поділ роботи між ПА, виявляє авторитетні джерела і створює N-агенти. Арбітр ресурсів перерозподіляє ресурси МАС між D-агентами для забезпечення еволюції шляхом добору та регулює кількість активних D-агентів.

Нові агенти даних створюються за генетичними алгоритмами через операцію кросоверу векторів генотипів агентів-батьків і випадкові зміни – мутації.

На початку сеансу пошуку створюється агент даних, що виконує попередню обробку вихідних даних: підрахунок частот різних словоформ, що зустрічаються в тексті; часткову морфологічну обробку шляхом усікання закінчень; вибір з набору схожих словоформ терміну – орту векторної моделі. Для побудови векторної моделі обираються терміни з частотою, більшої за певний поріг. Порогове значення визначається довжиною документа.

Передбачено використання таких ІПС, як Апорт, Рамблер і Яндекс; є можливість вибору кількості одержуваних результатів. Інтерфейс користувача дозволяє починати пошук за допомогою

ключових слів або вибору тексту-зразка. Обраний документ може бути використаний як зразок для наступного пошуку, при цьому користувач одержує список термінів документа.

Пошукова система "*Серверный Следопыт*" ЗАТ "МедиаЛингва" забезпечує пошук інформації на окремому Web-сайті або сервері корпоративної мережі. Основні функціональні можливості:

- можливість запиту природною мовою (російською й англійською);
- пошук за змістовною близькістю до запиту;
- ранжирування документів за ступенем близькості до запиту;
- виділення знайдених слів і виразів у документах;
- автоматичне формування анотацій знайдених документів.

Перш ніж робити пошук документів, система накопичує інформацію про місцезнаходження слів на сторінках Web-вузла і зберігає її у спеціалізованої БД, забезпечуючи надалі швидкий пошук. У комплект входять: сервер індексування; програма автоматичного реферування документів; пошуковий сервер; агент індексування, що передає інформацію на сервер індексування; пошуковий клієнт, реалізований у вигляді CGI-застосунків; програма керування компонентами.

Розподілена мультиагентна пошукова система *FireExpert* (http://uchcom.botik.ru/nut/fe.html) орієнтована на тих користувачів, що мають постійні інтереси в мережі (наприклад, пов'язані з їх захопленнями, роботою) і мають потребу в постійному надходженні інформації. Запити таких користувачів від одного сеансу пошуку до іншого практично не міняються, а користувачі є експертами в галузі пошуку. Засоби системи орієнтовані на передачу запиту користувача на кілька ПМ (таких, як Yahoo, AltaVista, Yandex) і фільтрацію результатів на основі заздалегідь складеного користувачем опису ПрО пошуку у формі онтології.

У системі використовується поняття образу релевантного документа, який будується на основі запиту користувача, виведення на онтології та евристичних правилах. Такий образ є описом характеристик релевантного запиту документа та описується через систему фреймів. Онтологія ПрО подається за допомогою неоднорідних семантичних мереж. Механізм виведення на онтології – це система продукційних правил, що містить блок розширення базового списку понять за рахунок використання властивостей відношень, що пов'язують їх з іншими поняттями онтології. Для конкретної ПрО визначають евристичні правила, що описують характерні риси HTML-документів з цієї області. На відміну від

більшості аналогічних систем запит у FireExpert вибирається зі списку доступних тем, а не вводяться через ключові слова або як фраза природною мовою.

Racing (Rational Agent Coalitions for INtelliGent Mediation of Information Retrieval on the Net – http://www.zsu.zp.ua/racing) – набір Web-сервісів, який здійснює раціональний інформаційний пошук [56, 57]. Система базується на агентних технологіях. В проекті Racing використовується методологія семантичного перетворення початкового пошукового запиту користувача у формі ключових слів або фраз до результуючого запиту, складеного з відповідних термінів онтології ПрО. Проект застосовує парадигму раціонального агентства (Rational Agency Paradigm – RAP) для розподілених, автономно створених, довільно структурованих та формалізованих ІР, що є частиною наступного покоління Semantic Web. У проекті для виконання пошуку використовуються онтології, подані у форматі DAML+OIL. Онтології як формальні теорії, що забезпечують розподіл та повторне використання знань, використовуються не тільки для анотування ІР, формулювання та розподілу запитів, але й для забезпечення семантичної інтероперабельності серед раціональних інтелектуальних ПА-посередників, які входять до складу коаліції. В рамках проекту розроблена методологія раціонального посередництва.

3.4. Мультиагентні інформаційно-пошукові системи

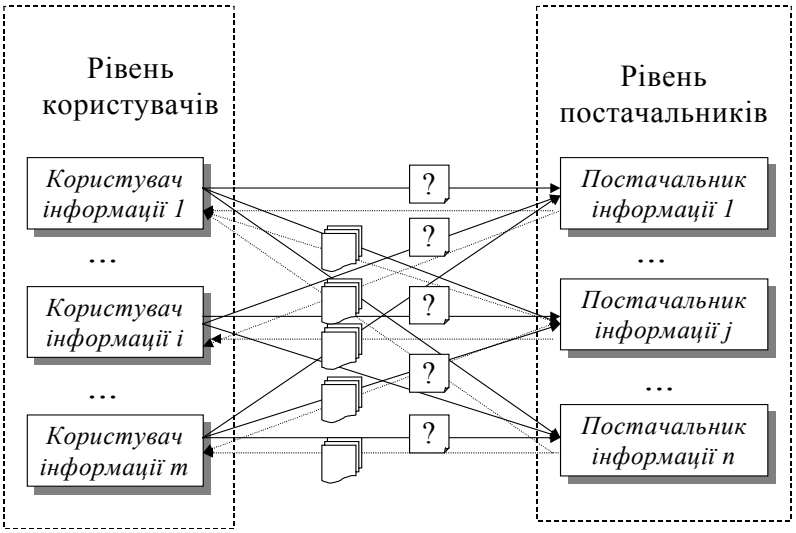
Використання мультиагентних систем для пошуку інформації

Розглянуті вище приклади показують, що для пошуку інформації в Інтернеті ефективним є використання МАС, у яких спеціалізовані ПА взаємодіють один з одним для спільного рішення поставленої користувачем цілі.

Інформаційний пошук полягає у взаємодії між користувачами та постачальниками інформації, в процесі якої у відповідь на інформаційний запит користувача постачальник інформації надсилає йому певні відомості, що релевантні цьому запиту.

Встановлення безпосередньої взаємодії всіх потенційних користувачів з усіма потенційними постачальниками призводить до низки проблем, пов’язаних з різноманіттям засобів подання інформаційних запитів та відповідей на них. У загальному випадку для m користувачів інформації та n її постачальників потрібно мати $m * n$ засобів перетворення подання інформації (рис.3.13). Крім того, з

появою нового постачальника треба поновлювати такі засоби для всіх користувачів.

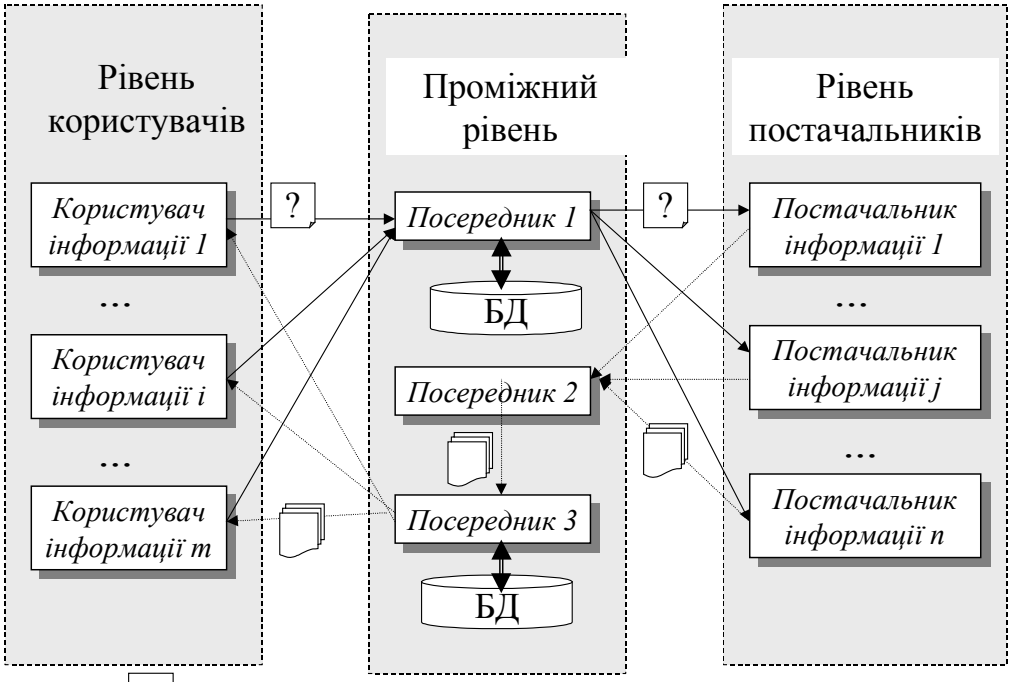


де ? –- інформаційний запит користувача,

–- результат пошуку інформації.

Рис.3.13. Модель безпосередньої взаємодії користувачів та постачальників інформації

Для ефективного обміну інформацією між споживачами інформації та її постачальниками необхідні посередники, які спрощують цю взаємодію. Розглянемо модель пошуку інформації, представлену на рис.3.14, яка має три рівні: 1) *користувачів* (тих, хто шукає інформацію), 2) *постачальників* (тих, хто надає інформацію) і 3) *проміжний*, що забезпечує зв'язок між першим і другим рівнями. Основна перевага такої моделі пошуку полягає у відсутності прямих зв’язків між користувачами та постачальниками інформації.



де ? – інформаційний запит користувача,

– результат пошуку інформації.

Рис.3.14. Трирівнева модель пошуку інформації.

На кожному з цих рівнів доцільно використовувати ІПА:

- для рівня користувачів задача агентів полягає в тому, щоб точно визначити, що саме шукає користувач, чого він хоче, чи є в нього які-небудь додаткові вимоги до цієї інформації тощо;
- для рівня постачальників задачею агентів є упорядкування точного опису інформації, яку надає цей постачальник, і відновлення цих даних при змінах контенту;
- для посередників задачею агентів є ефективна взаємодія між агентами першого і другого рівнів, тобто між користувачами і постачальниками інформації.

Архітектура мультиагентної інформаційно-пошукової системи

Розглянемо мультиагентну інформаційно-пошукову систему (МАІПС), .яка призначена для пошуку в Інтернеті документів, що релевантні запитам користувачів, у певній ПрО. Вона забезпечує ефективне використання ресурсів мережі, зменшує час пошуку, що витрачає користувач порівняно з використанням іншими ІПС, та підвищує якість пошуку.

Для постійної роботи користувача з ІР Інтернету характерна тривала зацікавленість в інформації, що відноситься до певної, досить вузької ПрО.

МАІПС забезпечує виконання складних багаторазових запитів в спеціалізованих ПрО, пов’язаних з професійними або науковими інтересами користувачів. Запити таких користувачів можуть повторюватися від сеансу до сеансу або змінюватися, але залишатися у рамках певної ПрО пошуку, в якій користувачі є експертами.

МАІПС не замінює собою ІПС. Вона є посередником між користувачем та існуючими засобами пошуку. Її, призначення – зробити звертання користувача до ІР Інтернету більш ефективним, зручним та швидким. Система містить ПА, які відповідають трьом рівням пошуку інформації: агентів користувачів, агентів ІР та агентів-посередників. Агенти МАІПС поділяються на два типи: прикладні і службові. Агенти ІР і агенти користувачів є прикладними, а агенти проміжного шару – службовими. Кількість службових агентів визначається їх функціями у МАІПС, а кількість екземплярів прикладних агентів не обмежена та визначається кількістю користувачів та ІР.

До складу МАІПС [342] входять:

- сервер з БД і відповідним ПЗ;
- аплети, що надаються користувачам МАІПС для автономної роботи (приміром, створення або редагування онтологій ПрО та постійних запитів);
- ПА для забезпечення функцій, пов'язаних з виконанням інформаційних запитів користувачів.

Сервер МАІПС забезпечує такі функції:

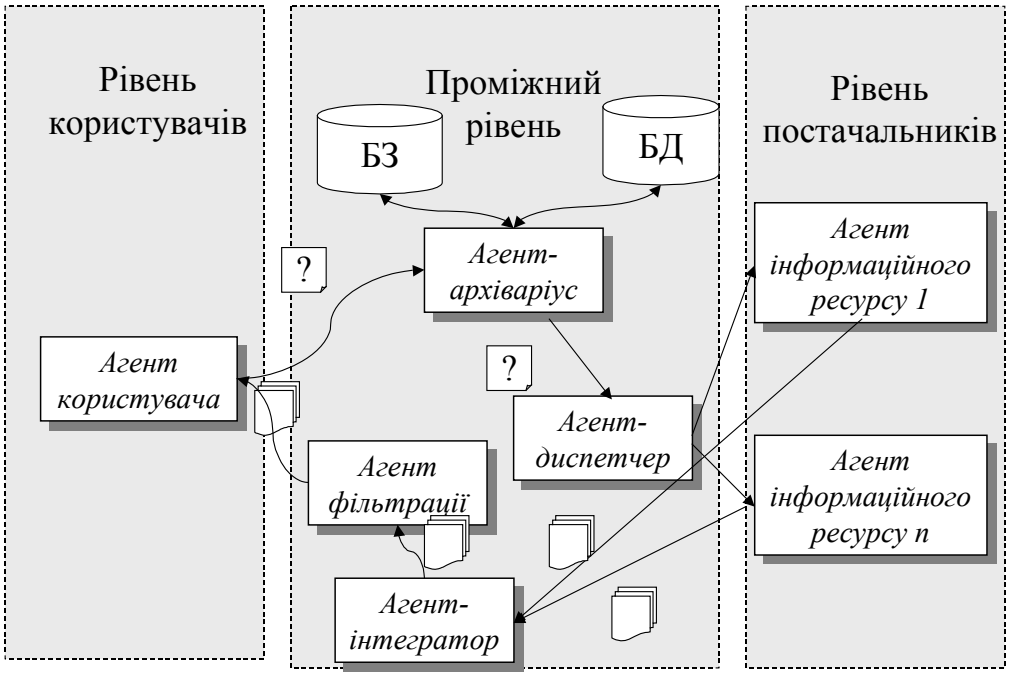
- реєстрацію користувачів, збереження створених ними онтологій, постійних інформаційних запитів і результатів виконання цих запитів;
- реєстрацію нових ІР ;
- виконання постійних інформаційних запитів користувачів і фільтрацію результатів виконання цих запитів;
- накопичення відомостей про виконання постійних запитів;
- збереження персональних даних користувачів МАІПС.

На *рівні користувачів* (рис.3.15) в МАІПС діють агенти, що пов’язані з конкретними користувачами.

Основні функції цих ПА:

- переформулювання запиту користувача, яке базується на персоніфікованій інформації про цього користувача та враховує попередній досвід його інформаційного пошуку;
- передання запиту іншим ПА;

отримання результатів запиту та подання їх користувачу у зручній для нього формі.



де ? – інформаційний запит користувача,

– результат пошуку інформації.

Рис.3.15. Архітектура МАІПС

На *проміжному рівні* в МАІПС діють службові ПА: агент-архіваріус, агент-диспетчер, агент-інтегратор та агент фільтрації. Основна функція цих агентів-посередників – встановлення ефективної взаємодії між ПА користувачів та ІР у виконанні запитів.

Агент-архіваріус отримує запит від агента користувача, порівнює його з записами у БД МАІПС, де накопичується інформація про виконані раніше запити та їх результати.

Якщо у БД знайдено саме такий запит, що вже виконано, тоді агент-архіваріус перевіряє актуальність його результатів, і, якщо вона є задовільною, передає отриманий результат агенту користувача. Якщо архіваріус не знайде у БД відповідного запиту, він передає цей запит агенту-диспетчеру для виконання.

Якщо ж у БД знайдено відповідний запит, але немає повідомлення про його результати (тобто запит ще виконується), агент-архіваріус передає агенту-диспетчеру повідомлення з рекомендацією підвищити статус запиту (якщо певна інформація потрібна значній кількості користувачів одночасно, швидке виконання такого запиту може підвищити ефективність роботи системи в цілому).

Агент-диспетчер обирає відповідних агентів ІР для виконання наданого йому запиту. Він отримує від агента-архіваріуса запит та, проаналізувавши його параметри та дані про ІР у БД МАІПС (мову, тематику, кількість запитів до цього ІР, які виконуються на поточний час, оцінку релевантності та актуальності результатів звертань до нього), передає цей запит агентам обраних ІР.

Функцією *агента-інтегратора* є об'єднання відповідей на запит, які надані різними агентами ІР, в єдиний список, що не містить повторів або неактуальних посилань. Цей список передається агенту фільтрації.

Агент фільтрації аналізує цей список на основі досвіду виконання попередніх інформаційних запитів користувача, відомості про які містяться в БД МАІПС, та контексту, в якому виконувався пошук, і відкидає посилання, які вважає не потрібними користувачеві. В аналізі використовуються онтології ПрО, яка цікавить користувача. Модифікований список посилань передається ПА користувача.

На *рівні постачальників інформації* в МАІПС діють *агенти ІР*. Кожному з них відповідає певне джерело інформації: глобальна або локальна ПМ, html-документ, що містить набір посилань на інші ІР тощо. Основною функцією цих ПА є трансформація запиту, який вони отримують через агентів-посередників від агентів одного або кількох користувачів, у формат, зрозумілий відповідному ІР, та його виконання. В результаті виконання запиту агент ІР створює список, який складається з адрес та анотацій знайдених ним документів, потім перевіряє доступність посилань та передає цей список агенту-посереднику, який відповідає за інтеграцію відповідей, наданих агентами різних ІР.

Відповідно до трирівневої моделі пошуку інформації, службові ПА можуть взаємодіяти один з одним і з прикладними ПА, проте прикладні ПА не можуть взаємодіяти один з одним безпосередньо.

Модель МАІПС складається з моделей класів ПА, які входять до її складу. Всі ці ПА мають подібну структуру, але відрізняються за функціями, які вони виконують.

Модель кожного класу агентів описується трьома моделями, які відповідають переконанням, цілям та бажанням агентів. Інтерфейс запиту у МАІПС показано на рис.3.16.

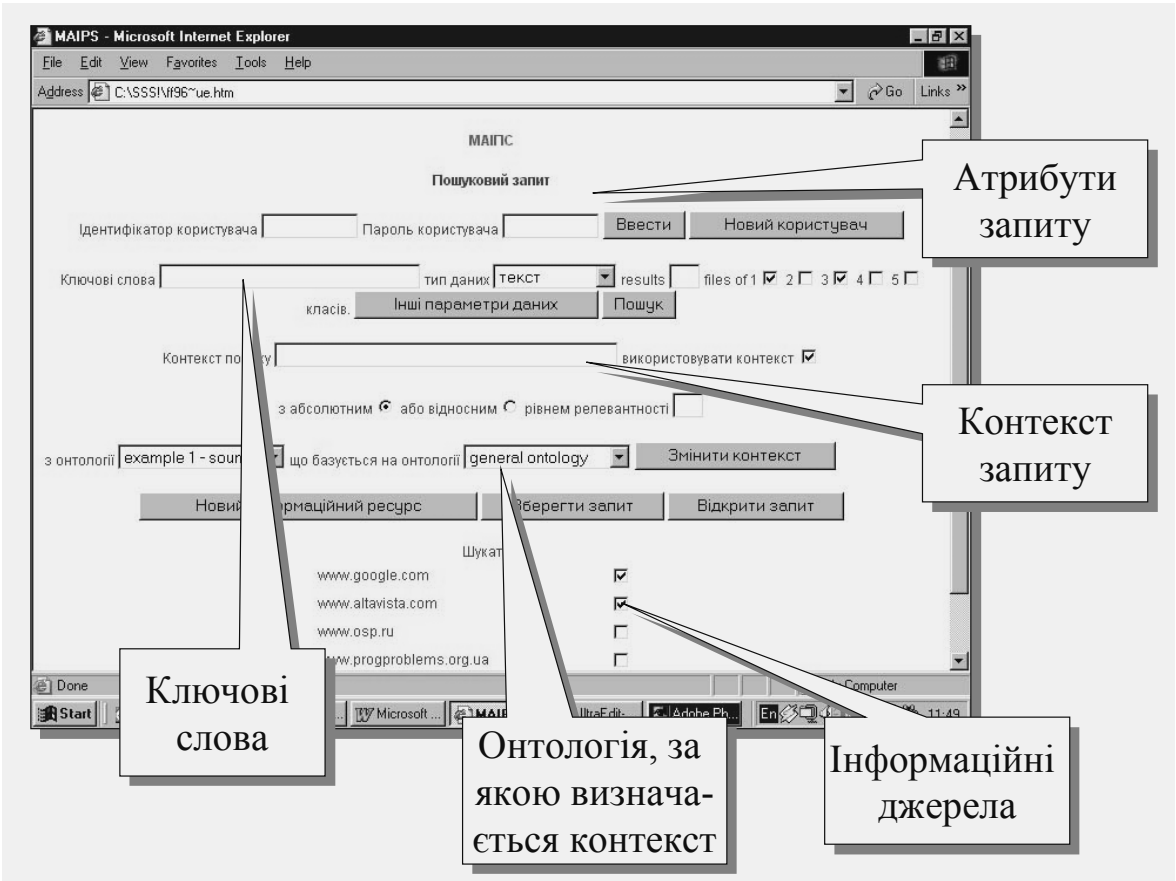


Рис.3.16. Запит користувача МАІПС

Для створення моделі МАІПС, відповідно з методологією розробки BDI-агентів, треба виконати наступні дії:

- визначити функції класів ПА, що входять до складу системи, та їх взаємодію;
- ідентифікувати тривалість життя ПА кожного класу та умови, за якими вони починають та припиняють свою діяльність;
- ідентифікувати інформаційний зміст взаємодії класів агентів; події й умови, що мають бути помічені; дії, що будуть виконуватися; інші інформаційні вимоги;
- побудувати класи агентів через успадкування або укрупнення, керуючись спільністю тривалості життя, інформації, інтерфейсів і послуг;
- описати для кожного класу ПА моделі їх переконань, цілей та намірів, навести конкретні класи та екземпляри агентів.

Моделі переконань класів агентів

Переконання агентів користувачів містять такі компоненти:

- онтологія ПрО, яка цікавить агента:
  - набір концептів ПрО;
  - набір відношень між концептами ПрО;

- набір правил виведення;

- переконання, пов’язані з різними ІР (спочатку апріорно задається користувачем, потім може динамічно змінюватися за допомогою індуктивного узагальнення досвіду ПА):
  - список заборонених ІР;
  - список бажаних ІР;
  - переконання, пов’язані з розкладом роботи;
  - переконання, пов’язані з мовами, які зрозумілі користувачеві (задаються користувачем, можуть змінюватися тільки користувачем – у випадку, якщо він вивчив нову мову або забув одну з тих, які знав раніше);
  - переконання, пов’язані з оцінкою релевантності документів запиту (формуються за допомогою індуктивного узагальнення досвіду, накопиченого ПА).

Переконання агентів ІР практично є знаннями і складаються з таких компонентів:

- формат запиту:
  - представлення логічного оператора поєднання;
  - представлення логічного оператора перетину;
  - представлення логічного оператора заперечення;
  - представлення словосполучення;
- інформація, яка присутня в списку посилань, який ІР надає у відповідь на запит, та формат цієї інформації.

Переконання агентів-посередників містять інформацію про користувачів (їх статус та оцінки релевантності, які вони дають різним інформаційним ресурсам) та інформацію про ІР (статичну і динамічну).

Кожен клас агентів-посередників має свою модель переконань.

Переконання агента-архіваріуса містять:

- статус користувача за ідентифікатором його агента (може бути змінено адміністратором МАІПС);
- інформацію про користувацькі запити, які виконувалися раніше або виконуються зараз в МАІПС;
- правило визначення пріоритету запиту (статуси користувачів, які надали однакові запити, сумуються для визначення пріоритету запиту).

Переконання агента-диспетчера містять:

- переконання, пов'язані з характеристиками ІР;
- правила побудови черги на обробку та виконання користувацьких запитів;

- інформацію про завантаженість ІР та розклад їх завантаженості, час доступу тощо;
- переконання, пов’язані з тим, які ІР відповідають запитам яких користувачів.

Переконання агента-інтегратора містять інформацію про те, які посилання є ідентичними (враховуючи інформацію з сайтів-дзеркал) та про те, як поєднувати інформацію, знайдену різними ПМ.

Моделі цілей класів агентів

Цілі агентів користувачів можна поділити на основні та допоміжні. Основні відповідають виконанню функцій агента користувача, а допоміжні дозволяють підвищити ефективність функціонування ПА.

Основні цілі:

- знайти інформацію (один документ або фіксовану кількість документів, усю доступну інформацію тощо), релевантну запиту користувача;
- знайти інформацію за найменший час;
- вчасно надати користувачеві інформацію, яку він ще не запитував, але може запитати;
- ефективно використовувати канали зв’язку.

Допоміжні цілі:

- здобути знання про запити користувача, які можуть допомогти у досягненні основних цілей (наприклад, узагальнити результати багаторазових запитів і видобути їх спільні характеристики, які користувач не вказує явно, щоб потім додавати їх до запиту автоматично);
- здобути нові знання ПрО, що цікавить користувача (приміром, спростити онтологію ПрО шляхом відкидання термінів, що ніколи не використовуються у пошуку або доповнити онтологію новими термінами та зв’язками між ними, здобутими з документів – результатів пошуку).

Цілі різних користувачів у більшості випадків конкурують одна з одною.

Цілі агентів ІР:

- надати відповіді на запити;
- виконати більше запитів за менший час;
- знайти релевантну та актуальну інформацію за кожним запитом;
- надати найбільший обсяг інформації за кожним запитом.

Протиріччя між цілями агентів різних ІР можуть виникати, коли для одного ІР створено кілька агентів, що забезпечують виконання різних запитів.

Цілі агентів-посередників:

- забезпечити ефективну роботу МАІПС;
- намагатися використовувати повторно результати запитів;
- зменшити середній час отримання відповіді на запит;
- зменшити гарантований час отримання відповіді на запит;
- забезпечити ефективне використання ІР;
- підвищувати релевантність відповідей запитам.

Цілі агентів-посередників не конфліктують одна з одною, тому що ці агенти спільно працюють для досягнення результату.

Моделі намірів агентів

Наміри агентів, які займаються пошуком інформації, у більшості випадків зводяться до виконання наданого запиту. Множина намірів кожного агента користувача складається з таких намірів:

- виконати надані користувачем запити;
- виконати заздалегідь багаторазові запити користувача;
- обробити наявну інформацію для підвищення ефективності своєї роботи.

Перша група намірів має найвищий пріоритет, а третя – найнижчий.

Кожний з агентів ІР має тільки один намір – виконати запит, а потім повернути його результати інтегратору.

Агент-архіваріус може мати два наміри відносно кожного запиту: передати його агенту-диспетчеру й потім повернути користувачеві отримані від інтегратора результати або знайти відповідний список документів у власній БД, не звертаючись до диспетчера.

Агент-диспетчер має наміри передати запит користувача до підмножини відомих йому агентів ІР, яка формується ним на основі його переконань та цілей. При цьому він має враховувати сумарну оцінку одночасного виконання різних запитів агентами ІР. Порядок виконання дій залежить від пріоритетів запитів, статусу користувачів та часу надання інформації, ефективного розкладу звертання до різних ІР тощо.

Наміри агента-інтегратора полягають у поверненні архіваріусу узагальнених результатів запитів. Порядок виконання цих дій теж залежить від пріоритетів запитів, статусу користувачів та часу надання інформації.

Засоби взаємодії агентів МАІПС

Агенти, що входять до складу МАІПС, використовують стандартні засоби, які дозволяють уніфікувати та формалізувати процес розробки інтелектуальних ІПА: KIF – для моделювання інформаційних джерел; KQML – для взаємодії між агентами та OWL – для моделювання ПрО. Префіксна версія числення предикатів першого порядку KIF [97] може бути використана як спільна мова змісту для агентів, що використовують різні мови представлення. KQML [75] – це одночасно формат повідомлень і протокол їх обробки для обміну знаннями між ПА. Використання цих стандартів надає можливість розширення МАІПС агентами інших розробників.

Реєстрація ІР в МАІПС

Для того щоб будь-яка ІС, у тому числі й МАІПС, отримала можливість працювати з ІР, необхідно, щоб цей ІР був у ній зареєстрований. У ПМ функції створення індексу виконують спеціальні агенти-кроулери, а в каталогах Інтернет – люди-експерти. У багатьох системах для реєстрації ІР необхідно, щоб власники ресурсу звернулися до власників системи з запитом щодо реєстрації та надали певні відомості про свій ІР. Іноді власники пошукової системи знаходять ІР самі (приміром, через інші ІПС), а потім будують його опис самостійно або звертаються до власників ІР з проханням заповнити певну форму для реєстрації.

Якщо ІР має метаопис (приміром, у форматі RDF), тоді його можна автоматично проіндексувати, здобуваючи опис його семантики з цього метаопису.

При реєстрації ІР у МАІПС спільними зусиллями розроблювачів ІР і МАІПС заповнюється відповідна форма, яка містить відомості про ІР і правила (формати, протоколи) взаємодії з ним, а сам ІР описується у вигляді онтології. При наявності RDF-опису ресурсу цей опис є основою для створення онтології ІР, яка надалі може бути доповнена за допомогою ключових слів і онтологічних термінів запитів, успішно виконаних за допомогою цього ІР.

Форма реєстрації ІР містить правила перетворення запиту користувача, яка складається з довільної кількості ключових слів $\langle t_1,...,t_n \rangle$, у подання, зрозуміле пошуковому механізму ІР (за відсутності власних пошукових механізмів здійснюється завантаження контента ІР для обробки засобами МАІПС).

Усі ІР, доступні МАІПС, можна розділити на дві категорії (рис.3.17):

- реєстрація здійснюється самостійно розроблювачами МАІПС;
- у реєстрації беруть участь розроблювачі ІР, які поміщають на сервері ІР зовнішній агент ІР для МАІПС.

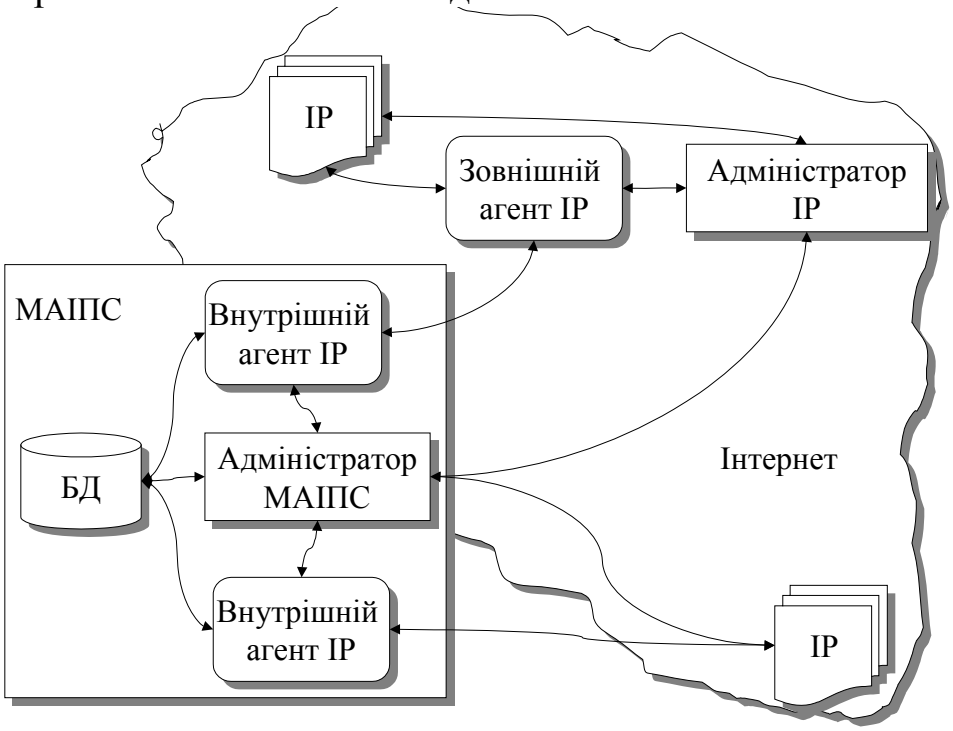


Рис.3.17. Реєстрація ІР у МАІПС

Індуктивне узагальнення як метод навчання агентів МАІПС за власним досвідом

Через складну структуру даних, серед яких здійснюється пошук, користувачеві досить важко самостійно задавати актуальні параметри пошуку. ІПА здатні допомогти йому в цьому, використовуючи методи індуктивного узагальнення, описані у главі 1. Ці методи дозволяють здобувати знання з тієї інформації, яку користувач отримує в процесі виконання пошуку.

Враховуючи метазнання про структуру цієї інформації та специфіку проблеми, яку вирішує МАІПС, можна заздалегідь вказати, яку саме інформацію доцільно індуктивно узагальнювати для модифікації переконань ІПА.

Для агентів користувачів як навчаючу вибірку слід використовувати відповіді на постійні запити. Роль параметра, за яким виконується класифікація, виконує оцінка релевантності відповіді запиту, надана користувачем. Для перетворення кількісних значень параметрів в якісні користувач має задати інтервали дискретизації або кількість значень, які слід використовувати.

Розглянемо це на прикладі такого обов’язкового параметру будь-якого документа, як його розмір. У першому випадку користувач задає, приміром, три інтервали: менше 10 Кб – "малий документ", від 10 до 50 КВ – "середній документ" і більш 50 Кб – "великий документ". Ці значення використовують в процесі індуктивного узагальнення. У другому випадку всі документи, що входять до навчаючої вибірки, тобто ті, що були запропоновані користувачу як результати виконання його запитів, сортуються за розміром, потім для першої третини списку значення розміру документа замінюється на "малий документ", для другої – на "середній документ", а для третьої - на "великий документ"".

Побудувавши класифікаційне правило, можна відкинути несуттєві параметри запитів (приміром, користувач кожного разу вказує, що його цікавлять документи, створені після 1980 року, а до цієї множини входять усі можливі документи) або додати параметри, за якими користувач насправді оцінює релевантність документа, не вказуючи їх явно у запиті (приміром, користувач знає тільки українську мову, але його постійні запити не містять будь-яких обмежень, що пов’язані з мовою документа). За допомогою алгоритму IID3M [339], описаного у главі 1, можна визначити кількість інформації про результат, яку містить кожний з параметрів.

Аналогічно агент-диспетчер може шляхом узагальнення інформації про результати запитів, що отримані від агентів різних IP, здобувати нові знання про ефективність використання цих ресурсів, а агент-архіваріус – визначати статус агентів користувачів та наданих ними запитів.

Для агентів IP доцільно узагальнювати інформацію про запити, які вони успішно задовольняли. У цьому випадку параметрами навчальної вибірки можуть бути терміни онтології користувача, які використовувалися у різних запитах, або персоніфікована інформація про користувачів, інформаційні потреби яких прагнув задовольнити агент певного IP.

Використання знань, отриманих шляхом індуктивного виведення, потребує попереднього підтвердження користувача. Якщо таке підтвердження отримано, ці знання слід відновлювати кожного разу, коли змінюється навчаюча вибірка, за якою вони побудовані, тобто враховувати результати виконання запитів.

Обробка контексту пошукових запитів

Традиційні механізми пошуку в Інтернеті, як правило, розглядають запити користувача на пошук інформації ізольовано один від одного і не враховують отримані раніше результати. Як вже було сказано раніше, значно підвищити ефективність пошуку дозволяє його *персоніфікація*, тобто використання відомостей про попередні запити конкретного користувача і сферу його інформаційних інтересів. Враховуючи контекст пошуку – інформацію про користувача, його інтереси і виконані раніше запити – можна отримувати більш релевантні результати. На сьогодні існує кілька різних підходів до формалізації такого контексту.

Більшість таких механізмів (виключення складає лише Northern Light) навіть не пропонують службу, яка повідомляла б користувачам про появу нових сторінок, що відповідають конкретним запитам.

Наприклад, у проекті Inquirus [99] інституту NEC Research Institute контекстна інформація задається явно у вигляді категорії даних, які хоче знайти користувач. Ця інформація використовується для вибору механізмів пошуку, яким передається запит, для модифікації запитів і для визначення принципів упорядкування отриманих документів.

Як показує аналіз публікацій, один з перспективних підходів до завдання контексту пошуку ґрунтується на онтологіях, що містять перелік основних термінів, зв'язки між ними і правила виведення (так, у проекті Semantic Web, спрямованому на аналіз семантики IP , саме онтологічний підхід є основою для подання знань ПрО).

Проблема інформаційного пошуку ускладнюється тим, що різні групи людей, які займаються збиранням і пошуком інформації, використовують для спілкування з ІПС як свої спеціальні терміни, так і терміни, широко використовувані іншими співтовариствами в іншому контексті. Поряд із глобальними онтологіями, що описують досить широкі ПрО і для створення яких необхідні значні зусилля як експертів ПрО, так і інженерів зі знань, існують онтології, які дозволяють формально представити знання конкретного користувача ПрО. Такі онтології можуть створюватися і модифікуватися користувачами самостійно. Деякі переконання користувача щодо ПрО можуть бути помилковими, але така онтологія адекватно описує інформаційні потреби саме цього користувача (наприклад, якщо користувач помилково вважає дельфіна рибою і, запросивши

зображення якої-небудь риби, отримає зображення дельфіна, тоді його інформаційна потреба буде задоволена).

Для задання контексту пошуку створена користувачем може використовуватися онтологія ПрО.

В онтології користувач відзначає терміни, наявність яких у шуканому документі є бажаною (чи небажаною), а також задати більш складні операції – наприклад, автоматично відзначити всі терміни, що знаходяться в заданому відношенні з термінами, відзначеними раніше. Це дозволяє, зокрема, легко враховувати при пошуку синоніми чи близькі за значенням слова (приміром, "*виведення за аналогією*" та "*традуктивне виведення*"), слова, які мають кілька поширених варіантів написання (приміром, "*мультиагентна система*" та "*багатоагентна си*стема"), а також здійснювати пошук відразу кількома мовами.

За відзначеними в онтології термінами формується непорожня множина слів (або словосполучень) $W = \{w_1,...,w_m\}$, кожне з який має свою позитивну або негативну вагу $v_k, k = \overline{1,m}$. Для кожного документа $i_j, j = \overline{0,k}$ з множини $I`, I` \subseteq I$ формується коефіцієнт відповідності контексту пошуку $s_j, j = \overline{0,k}, s_j = \sum_{k=1}^{m} v_k * f(i_j, w_k)$, де $f(i_j, w_k) = \begin{cases} 1, \text{если } w_k \in i_j \\ 0, \text{если } w_k \notin i_j \end{cases}$.

Чим вище цей коефіцієнт, тим, імовірно, вище релевантність відповідного документа запиту користувача. Іноді доцільно використовувати більш складну формулу розрахунку коефіцієнта відповідності контексту пошуку $s`_j, j = \overline{0,k}, s`_j = \sum_{k=1}^{m} v_k * f(i_j, w_k) * t_k$, де $t_k, k = \overline{1,m}$ – кількість входжень терміна $w_k, k = \overline{1,m}$ в документ $i_j, j = \overline{0,k}$.

Щоб створити онтологію, користувач має задати скінченну множину термінів ПрО X, скінченну множину відношень між цими термінами $\Re$ і скінченну множину функцій їх інтерпретації Φ, а потім указати, між якими саме термінами існують які відношення (рис.3.18). Після цього користувач створює свій інформаційний запит та задає за онтологією контекст пошуку. ІПС трансформує цей запит з урахуванням контексту пошуку, обирає потрібні користувачу ІР та передає йому відомості про ці ІР. Спосіб виконання пошуку залежить від специфіки конкретних ІР.

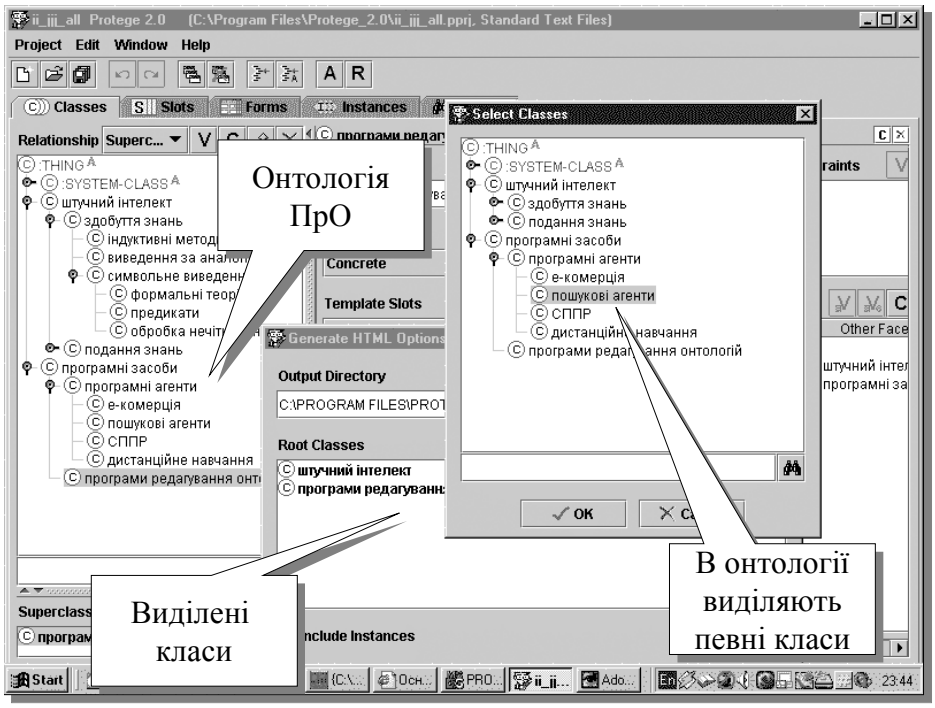


Рис.3.18. Подання онтології ШІ в Protégé

Для того щоб користувач мав можливість приступити до інформаційного пошуку, йому треба надати непорожню множину ІР Q, $Q = \langle Q_1,...,Q_n \rangle$, до яких він може звернутися. Такими ІР можуть бути глобальні і локальні ПМ, окремі сайти, фіксовані документи тощо.

У результаті виконання пошуку формується множина документів I, які ІПС вважає релевантними запиту.

$I = \bigcup_{i=1}^{n} I_j$, де I_j – результат пошуку в ІР Q_j.

Якщо є метаінформація про відповідний ІР (наприклад, у форматі RDF або MPEG7), то пошук здійснюється з її урахуванням.

На жаль, більшість ІПС, що здійснюють пошук за ключовими словами, включають у I дуже багато непотрібної користувачу інформації: повторювані, нерелевантні і застарілі посилання, а також посилання на документи, уже йому вже відомі. Щоб позбавити користувача від необхідності переглядати вручну всі ці документи, потрібно здійснити їхню фільтрацію, використовуючи відомості про попередні запити і сферу інформаційних інтересів користувача.

Етапи обробки результатів виконання запитів (рис.3.19):

1. У результаті виконання інформаційного запиту користувача до ІР Q за ключовими словами формується множина I.
2. Якщо множина I не порожня, то виконується її упорядкування за URL-адресами посилань. Інакше – завершення роботи.

3. Якщо отримана на кроці 2 підмножина I не порожня, то відфільтровуються посилання-“дзеркала”. Інакше – завершення роботи.
4. Відфільтровуються застарілі посилання.
5. Якщо отримана на кроці 3 підмножина I не порожня, то перевіряється за допомогою БД, чи одержував користувач раніше ці посилання (якщо одержував, тоді рішення про те, що робити з посиланнями, залежить від того, як у минулому користувач обробив це посилання, а також від інших його інструкцій). Інакше – завершення роботи.
6. Якщо сформована на кроці 5 підмножина I не порожня, то перевіряється відповідність документів $i_j, j = \overline{0,k}$ з множини $I`, I` \subseteq I$ контексту пошуку. Інакше – завершення роботи.

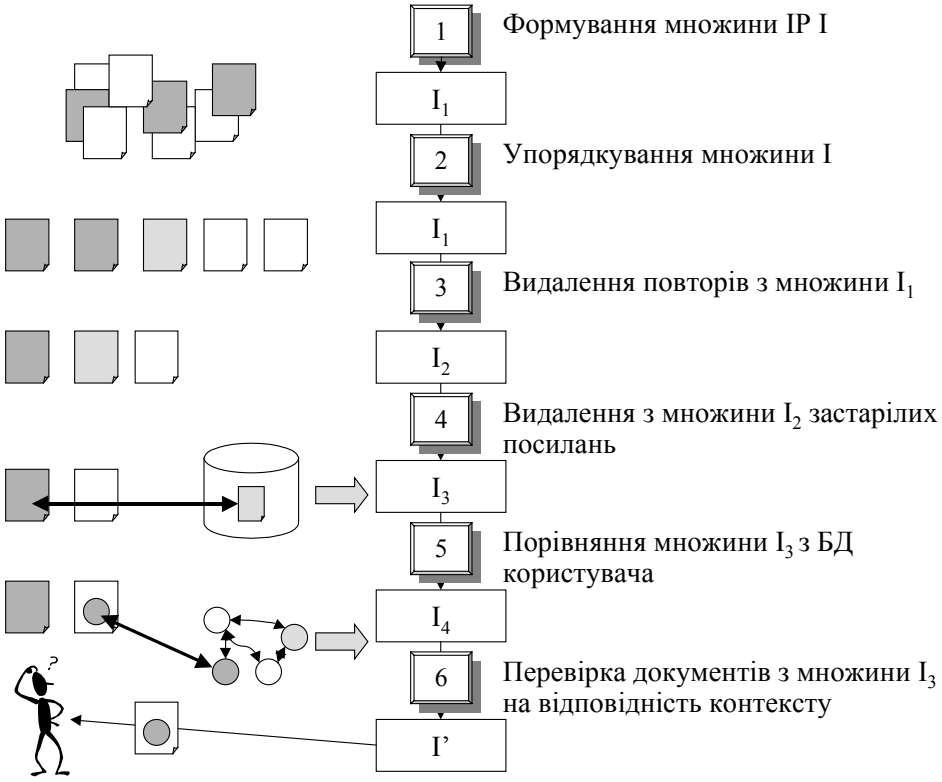


Рис. 3.19. Етапи обробки результатів запитів

Саме на 6-му етапі використовується онтологія ПрО, створена раніше користувачем. Застосування онтологій для завдання контексту пошуку, як зазначено вище, орієнтоване на тих користувачів, що мають сталі інформаційні інтереси в мережі і потребують постійного надходження відповідної інформації (рис.3.20).

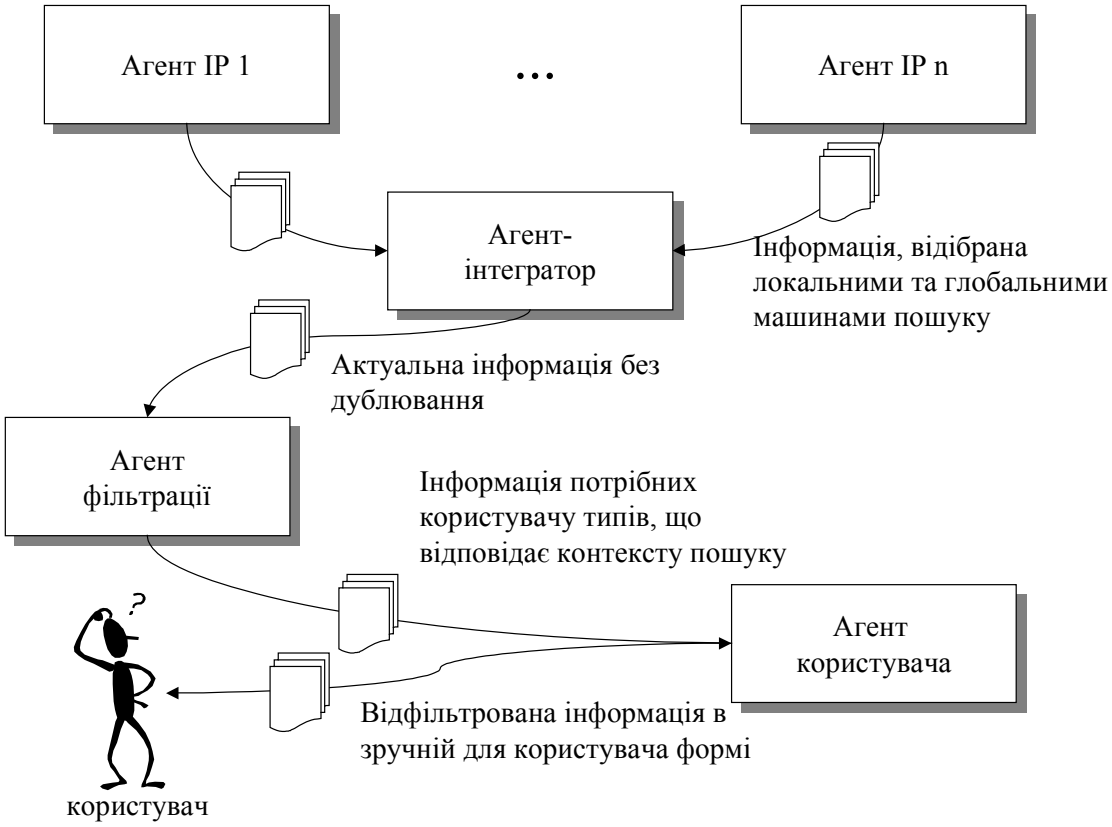


Рис.3.20. Процес виконання запиту користувача

Запити таких користувачів можуть повторюватися від сеансу до сеансу чи змінюватися, але обмежена ПрО пошуку, у якій користувачі є експертами, практично не змінюється. Опис цих ПрО задається самими користувачами у виді онтологій, що містять перелік основних термінів, зв'язки між ними і правила виведення. Один користувач може створювати кілька онтологій, якщо він має кілька цікавлячих його прикладних областей, що не перетинаються.

Користувач може звертатися до онтологій, створених іншими користувачами: переглядати їх, задавати за ними контекст пошуку, копіювати з них потрібні фрагменти, але не має права змінювати їх. ІПС має передбачати пошук онтологій, що містять уведені користувачем терміни, а також пошук онтологій, схожих на обрану користувачем. Це дозволяє створювати групи користувачів із спільними інформаційними інтересами і запобігати дублюванню у виконанні однакових багаторазових запитів різних користувачів.

Висновки

Проаналізувавши ІР, доступ до яких користувачі отримують через Інтернет, засоби їх подання та індексації, можна зробити

висновок про доцільність розробки агентно-орієнтованого ПЗ, що використовує знання ПрО, подані у стандартизованому виді, та враховує персоналізовані відомості про користувачів. Основу для такої формалізації сьогодні надають онтологічні системи та метаописи.

Для прогнозування поведінки мультиагентної інформаційно-пошукової системи необхідно створити її теоретичну модель. Ефективним засобом для цього є використання таких ментальних понять, як цілі, наміри та переконання.

Щоб система могла адекватно реагувати на зміни у своєму інформаційному оточенні, необхідно надати їй засоби узагальнення власного досвіду на основі алгоритмів індуктивного та традуктивного здобуття знань. Це приведе до більш інтелектуальної поведінки й отримання більш релевантних результатів.

Глава 4. Застосування агентних технологій у *e*-бізнесі

Те, що ми називаємо прогресом, являє собою заміну однієї неприємності іншою

Г.Елліс

Інформація слугує визначальним чинником розвитку міжнародної, економічної, технічної та наукової сфер людської діяльності. Саме інформація забезпечує радикальну інтенсифікацію суспільного виробництва на базі використання ІТ, зростання продуктивності праці, інтелектуалізацію праці.

Перетворення інформації в товар стимулює впровадження передових ІТ у практику економічної і господарської діяльності. Економіка стає усе більш інформаційно ємною. У XXI столітті інформаційна складова економічної взаємодії стає визначальною.

Економічна інформація - сукупність будь-яких даних, що відображають явища економічного стану суспільства. Економічна інформація у сфері матеріального виробництва є інструментом керування виробництвом та економікою [330].

Автоматизація процесів обробки різних інформаційних об’єктів призводить до формування сховищ даних, БЗ, інформаційних фондів, каталогів, тезаурусів, онтологій тощо. Знання, що використовуються в інформаційній економіці, класифікують на *базові*, *специфічні* і *інноваційні*. Базові знання надають необхідний мінімум для входження суб'єкта підприємництва в інформаційну економіку. Специфічні знання забезпечують конкурентний потенціал суб'єкта підприємництва, надаючи йому певні переваги порівняно з аналогічними підприємствами. Інноваційні знання надають фірмі можливість змінювати сам процес функціонування. Саме вони і дають підприємству можливість бути лідером у певній галузі.

4.1. Інформаційна економіка як сфера застосування агентних технологій

Сукупність даних, що стосуються певної ділянки управлінської роботи, називають інформаційним потоком. *Інформаційна економіка* (*і*-економіка) – новий науково-практичний напрямок економіки, пов'язаний з дослідженням руху інформаційних потоків, поданих у електронній формі, в економічних системах. Інформаційна економіка є базисом для таких форм ділової активності, як електронний бізнес,

електронне урядування, електронний документообіг тощо. І-економіка – це інформаційна модель реальної економіки.

Процеси, що підсилюють роль і-економіки [322]:

- функціонування інформаційно-комунікаційного середовища, яке створює глобальна мережа Інтернету;
- наявність економічних агентів і інфраструктури, що уможливлює їхню діяльність;
- використання суб'єктами бізнесу глобальних телекомунікаційних мереж як інструменту для реорганізації їхньої спільної діяльності, тобто створення мережної організації, яка не може бути віднесена ані до командно-адміністративної, ані до ринкової форми;
- впровадження Інтернет-технологій, що приводить до модернізації інфраструктури в економіці і створення мережних інституціональних структур.

Застосування ІТ-технологій у сфері бізнесу викликало появу нової форми взаємодії зі споживачем – електронного бізнесу (*е*-бізнесу). *Електронний бізнес* – комерційна діяльність, що базується на інформаційно-комунікаційних технологіях та спрямована на отримання прибутків.

Розвиток *е*-бізнесу пов'язаний з формуванням єдиного економічного простору, впровадженням корпоративного електронного документообігу, застосуванням електронного цифрового підпису та електронних угод, персоніфікацією доступу до економічної інформації і впровадженням інтелектуальних механізмів підтримки бізнес-процесів. Агентні технології – один з ключових напрямків розвитку інформаційної економіки. Відповідно до [200] *єдиний інформаційний простір* (ЄІП) – сукупність інформації, технологій її використання та засобів передачі, що функціонують на основі єдиних принципів і за спільними правилами для задоволення інформаційних потреб користувачів. Він містить систему сервісів, найважливіші з яких – це:

- *обчислювальні сервіси* (Data/Computation Services) – засоби розміщення даних і їхнього транспортування між застосунками, доступу до обчислювальних і мережних ресурсів**;**
- *інформаційні сервіси* (Information Services) – засоби подання, обробки, збереження та доступу до інформації**;**
- знання-орієнтовані сервіси (Knowledge Services) – засоби накопичення, подання, відновлення, публікації знань.

Сервіс – програмний процес, що реалізує дію застосунку в певній ПрО. Архітектура .Web-сервісів складається з трьох компонентів – провайдера, користувача та брокера.

Базові операції Web-сервісів – це публікація, пошук і зв'язок. Провайдери сервісів публікують свої сервіси через брокерів сервісів, використовуючи протокол WSDL [237], заснований на XML. Користувачі можуть шукати необхідні сервіси серед опублікованих у каталозі за відкритим протоколом UDDI. Сервіси можуть програмно викликатися через Інтернет. ПА в цій схемі можуть виступати у ролі посередників на ринку інформаційних послуг: агент з власної ініціативи або за дорученням іншого агента організовує пошук потрібного сервісу в певному репозиторії, перевіряє повноваження користувача та запускає сервіс у роботу.

Охарактеризуємо основні процеси, які пов’язані з впровадженням інформаційної економіки.

Першим процесом, що створює технічні умови для формування і розширення масштабів інформаційної економіки, є розвиток і поширення ІТ, зокрема, Інтернет-технологій. Ці технічні засоби постійно розвиваються, а постійне зниження цін на їх придбання і використання підвищує доступність Інтернет-технологій.

Широко застосовують засоби групової роботи географічно розподілених учасників спільної діяльності (workflow, groupware), що дозволяє заощаджувати кошти, пов'язані з територіальним переміщенням людей, а також технології ШІ, у тому числі – агентні технології, які забезпечують ефект постійної присутності в Інтернеті суб’єктів економічної діяльності. Ці технології дозволяють знизити інформаційне перевантаження учасників інформаційної економіки, підвищити швидкість і ефективність процедур установлення контактів, проведення переговорів, підтримки угод тощо.

Другий процес формування інформаційної економіки пов’язаний з перенесенням до Інтернету різних видів соціально-економічної діяльності. Це стимулюється багатьма міжнародними і національними організаціями (приміром, у 1998 р. Всесвітня торгова організація прийняла рішення звільнити від обкладання митними зборами дані та ПЗ, придбані та доставлені через Інтернет, а Комісія Європейського Співтовариства затвердила Шосту Рамкову Програму з розвитку науки та технології на 2002-2006 роки для створення сприятливих умов для використання переваг Інтернет-технологій бізнесом, малими та середніми підприємствами та приватними особами у Європі – www.cordis.lu).

Третім процесом формування і-економіки є модернізація традиційних організацій у мережні структури на всій ієрархічній вертикалі економіки – від окремих фірм до об'єднуючих їх фінансово-промислових груп, корпорацій, ринків тощо. Цей процес пов'язаний з активним використанням ІТ, у тому числі і агентних технологій, і зменшенням частки традиційних ієрархічних форм керування. Модернізація організацій у мережні структури дає потенційні такі переваги:

- відсутність потреби у фізичному переміщенні учасників спільної діяльності (віртуальних робочих колективів) призводить до економії часу та витрат;
- ЄІП підприємства, корпорації, країни, регіону полегшує та прискорює доступ до створених ними ІР та дозволяє знизити витрати на формування і підтримку внутрішнього інформаційного середовища організації;
- колективне формування ІР дозволяє співробітникам оперативно отримувати інформацію про поточні бізнес-процеси та впливати на них.
- розвиток засобів координації та кооперації спільної діяльності підвищує точність прийнятих рішень, змінює структуру витрат підприємства: дешевше передавати на виконання роботи тимчасовим працівникам або зовнішнім компаніям (аутсорсинг), ніж тримати для цього штатних співробітників.

Четвертий процес формування і-економіки – створення мережних варіантів "горизонтальних" економічних структур, що обслуговують все різноманіття організацій в економіці. До таких структур належать: торгова та фінансова інфраструктури, система трудових відносин, юридична система тощо. Основними діючими елементами горизонтальних структур є зв'язки між їх окремими ланками і єдині правила роботи всіх ланок. Горизонтальні структури в економіці – мережі зв'язків, перенесення яких до Інтернету підвищує їх ефективність.

Сучасна комерційна діяльність здійснюється під керівництвом людини, яка вирішує, що саме та коли купити або продати, яку кількість товару і за якими цінами. Кожна людина має власний досвід, переконання та цілі, які визначають суб'єктивний характер соціально-економічної діяльності.

Соціально-економічні ситуації належать до класу слабоструктурованих та не повністю формалізованих областей. Однією з важливих особливостей цих ситуацій є необхідність враховувати

поведінку суб’єктів, які беруть участь у цьому процесі. Соціально-економічна ситуація містить елементи когнітивного конфлікту, в якому кожна сторона має власні уявлення про ситуацію та прагне змоделювати поведінку іншої, тобто конструює власні образи та партнерів. Для аналізу таких ситуацій використовують логіку рефлексивного керування [303]. При когнітивному моделюванні ситуацій її загальна модель – це когнітивна карта, яка може бути подана у вигляді орієнтованого навантаженого графа, вершини якого – фактори, що впливають на ситуацію, а дуги навантажені нечіткими значеннями взаємовпливаючих факторів [354]. Математичний апарат, що базується на нечітких когнітивних картах, дозволяє здійснювати аналіз соціально-економічної ситуації та синтезувати стратегії керування. Практичне використання рефлексивних моделей Лефевра обмежене можливостями їх математичного апарату. Для моделювання соціально-економічних ситуацій доцільно застосовувати агентний підхід.

Застосування агентних технологій в *e*-комерції пов’язане з поданням клієнта і продавця через ПА, що дозволяють створювати адаптивні моделі поведінки покупців і продавців на основі МАС.

ПА забезпечують встановлення та підтримку рівноправних зв’язків між економічними об’єктами інформаційної економіки.

Апаратом моделювання та інструментом дослідження *i*-економіки є агентний підхід щодо моделювання економіки, орієнтований на дослідження економічних систем, що складаються з множини розподілених застосунків, діючих паралельно без глобального контролю і відповідального за поведінку цих застосунків. Основна особливість цього підходу – дослідження того, як економічний порядок і регулярність виникають з локальних взаємодій автономних ПА, регульованих через параметри середовища життєдіяльності. Саме децентралізована економічна діяльність рівноправних ПА, координована за допомогою інформаційної імітації ними своїх дій за допомогою інформаційної моделі середовища їх існування, якнайкраще відображає бізнес-зв’язки об’єктів економічної системи.

Обмеженим ресурсом є те, заради чого ПА встановлюють між собою ці зв'язки: місце в системі розподілу праці, в якому даний агент становить для економічної системи максимальну цінність і, отже, одержує від участі в спільній діяльності максимально можливу для себе вигоду. Для того, щоб в групі окремий ПА мав змогу "усвідомити" можливість своєї участі в її діяльності і "передбачати"

можливі вигоди і ступінь зацікавленості групи в його діях, він повинен мати достатньо повну картину про можливості і наміри решти ПА. Урахування цієї обставини може бути реалізоване включенням у модель поведінки агента його BDI-моделі, яка містить інформаційні образи інших об’єктів, а також образи доступних йому фрагментів економічної системи.

Ментальна модель потрібна агентові, щоб програвати свої можливі дії і приймати рішення, але вона буде працездатна для забезпечення властивого громадським формам високого рівня взаємозалежності дій незалежних і рівноправних агентів тільки за існування механізму безперервної підтримки її в актуальному стані. Для цього інформаційні образи агентів-партнерів у ментальній моделі повинні мати зв’язки з реальними партнерами, які в умовах групи формуються в носії ментальної моделі через прямі контакти з партнерами. У цьому разі ментальна модель – колективно підтримувана субстанція, яка вже не належить одноосібно окремому агенту, хоча вона і може існувати тільки в його свідомості. Починаючи з певного рівня розвитку ІТ, ментальна модель може бути відчужена від агента на певному інформаційному носії, після чого вона вже є одним з об'єктів середовища ПА.

За рахунок активної спільної діяльності всього співтовариства ПА в актуалізації ментальної моделі можна вважати, що ця модель, тобто інформаційний образ середовища, є системою, що складається з ментальних моделей членів даного співтовариства, яка є більшою за їх об’єднання.

Для процесу координації на базі прямих зв'язків між агентами характерно, що вони "обговорюють" свої дії між собою, "домовляються" про взаємоприйнятні дії і потім "координують" свою спільну діяльність, за якою досягають домовленості. З урахуванням гіпотези про існування інформаційного образу середовища цей процес координації може бути уточнений: ПА відчужують у даний інформаційний образ свої пропозиції з приводу нових варіантів діяльності та нових комбінацій зв’язків. Сукупність таких пропозицій становить множину вибору кожного ПА системи. Агенти оцінюють прийнятність існуючих пропозицій і вносять свої, які також оцінюються рештою агентів. Якщо співтовариство агентів зафіксувало в своїй множині вибору взаємовигідний варіант, то він реалізується, а встановлені між агентами зв'язки використовуються для поточної координації діяльності.

В економіці прямих рівноправних зв'язків існує два підпростори діяльності ПА:

- *матеріальний* – реальні процеси створення, розподілу і споживання ресурсів;
- *інформаційний* – результат ментального (психологічного) відображення першого і включає процеси формування інформаційного образу середовища, а також колективне конструювання агентами на цій основі нового образу матеріального простору.

ПА створюють в інформаційному просторі образ матеріального простору і потім перебудовують його відповідно до поточних потреб. У цьому є певна циклічність: інформаційні образи нових зв'язків і видів діяльності, що ініціюються в другому підпросторі, частково матеріалізуються в структурі першого, міняючи його поточний стан; з другого боку, новий стан першого простору стає базисом для генерації нових станів і інформаційних образів, що заповнюють другий простір. Зв’язки ПА також можуть бути поділені на два типи: 1) обміну ресурсами, 2) обміну інформацією.

Найбільший економічний ефект від впровадження Інтернет-технологій пов’язаний з такими економічними інститутами, як торгівля, фінанси та трудові відносини.

Ефективність систем *е*-комерції, у тому числі й агентно-орієнтованих, визначає міру відповідності комерційним потребам її суб’єктів використаних у ній технологій, підходів, моделей і правил. Одним з важливих аргументів на користь впровадження агентних технологій у *е*-бізнесі є їх ефективність, але доведення цього потребує використання відповідних методик, які дозволяють порівнювати різні бізнес-застосунки.

Зараз багато методик оцінки ефективності Інтернет-проектів, до яких належать системи *е*-комерції, базуються на таких показниках роботи, як частота відвідуваності сайту і час, що проводить відвідувач на сайті. Наприклад, для електронного магазина важливим фактором оцінки ефективності його функціонування є кількість відвідувачів. Але ці показники не завжди відображають реальну ефективність застосунку. Приміром, на непрофесійно розробленому сайті користувач змушений провести значно більше часу для здійснення покупки, ніж на сайті зі зручною навігацією і зрозумілим інтерфейсом.

Існують такі взаємозалежні напрямки оцінки ефективності систем *е*-комерції:

- економічне;
- організаційне;
- маркетингове тощо.

Для порівняння систем *е*-комерції можна використовувати методи аналізу відкритих систем, які дозволяють виявити пріоритети осіб, що приймають рішення (ОПР), і визначення інтенсивності взаємодії компонент, які описують структуру системи ієрархії.

Застосування методу аналізу ієрархій для порівняння систем *е*-бізнесу [327] дозволяє виявляти пріоритети ОПР, і вибрати згідно з цими пріоритетами найбільш придатну систему. Метод аналізу ієрархій Сааті [348] полягає у декомпозиції проблеми на простіші складові і подальшій обробці послідовності суджень ОПР за парними порівняннями. У результаті може бути встановлено відносний ступінь взаємодії елементів ієрархії. Метод містить процедури синтезу множинних суджень, отримання пріоритетності критеріїв і знаходження альтернативних рішень. На першому етапі виявляють найважливіші елементи проблеми, на другому – найкращий спосіб перевірки спостережень і оцінки елементів; наступним етапом може бути розробка способу застосування рішення й оцінка його якості.

4.2. Програмні агенти *е*-комерції

Модернізація торговельної інфраструктури за допомогою інформаційно-комунікаційних технологій одержала назву "електронна комерція". У сучасній літературі часто використовують визначення UNCTAD (United Nations Committee on Trade and Development): електронна комерція – угоди, пов'язані з комерційною діяльністю організацій і фізичних осіб, що базуються на обробці і передачі інформації в електронному вигляді, у тому числі текстів, звуків і візуальних даних.

Електронна комерція (*е*-комерція) – технологія, що забезпечує повний замкнутий цикл бізнес-операцій, який включає замовлення товарів і послуг, проведення платежів, доставку товару, що проводяться з використанням електронних засобів та інформаційно-комунікаційних технологій і забезпечують передачу прав власності одних юридичних або фізичних осіб іншим [329].

Застосунки *е*-комерції поєднують мобільні системи та ШІ, які раніше розвивалися окремо.

Е-комерція може здійснюватися:

- між суб'єктами підприємництва в процесі виробництва і продажу товарів (модель *бізнес-бізнес*, B2B);
- між суб'єктом підприємництва та споживачем під час продажу і поширення товарів (модель *бізнес-споживач*, B2C);
- між споживачами (модель *споживач-споживач*, C2C)
- між державними установами та суб’єктами підприємництва (модель *установа-бізнес*, G2B) тощо.

Технології *е*-комерції змінили традиційний шлях здійснення бізнесу. Складність транзакцій збільшилася завдяки величезній кількості доступної інформації та динамічності навколишнього середовища. ПА здатні виконувати комерційні транзакції від імені користувачів. Це позбавляє людей від рутинних дій.

На сьогодні інтелектуальні агенти широко застосовуються для порівняння умов продажу товарів у магазинах для організації індивідуального обслуговування замовника, без втручання людини в процес взаємодії із замовником. ПА здатні навчатися з часом: вони запам'ятовують переваги покупця, його традиційні шаблони пошуку, зроблені ним раніше покупки – все це направлено на поліпшення обслуговування замовника.

Ролі ПА е-комерції

ПА *е*-комерції – це програмні сутності, що забезпечують процес переговорів між споживачами та продавцями та всі інші фази купівлі-продажу електронним способом. Кожен ПА має оцінювати можливі торговельні угоди, враховуючи переваги, які його користувач надає різним параметрам угоди. Агентів *е*-комерції можна характеризувати, враховуючи такі властивості [127]:

- роль, яку виконує ПА;
- раціональність поведінки;
- засоби подання та використання знань;
- форми зобов'язань, які підтримує ПА;
- здатність ПА до суспільної поведінки;
- стратегії визначення ціни товару або послуги.

Агенти використовують знання про товари та послуги, для яких вони встановлюють ціну, і про те, як інші ПА оцінюють подібні товари. Залежно від того, які правила поведінки задані користувачем, агент обирає стратегію, за якою пропонує продавцю ціну, яку він згоден сплатити. У процесі визначення ціни важливим чинником є персоніфіковані знання про користувачів.

ПА можуть виконувати в процесі переговорів ролі як клієнтів, так і продавців (або обидві ролі водночас). ПА покупців можуть автоматично збирати інформацію про продавців і товари та оцінювати різні пропозиції. У переговорних процесах ПА клієнтів і продавців – основні діючі об'єкти, але в деяких сценаріях (приміром, в аукціонах) головна роль належить агентам-посередникам.

ПА як посередники *е*-комерції в контексті загальної моделі маркетингових досліджень поведінки покупця CBB (Consumer Buying Behavior Model), яка охоплює дії та рішення, пов'язані з покупкою та використанням товарів і послуг. Модель CBB охоплює шість стадій, які дозволяють формально розподіляти за категоріями системи *е*-комерції [217]:

- *ідентифікація потреби* – на цій фазі покупець отримує детальну інформацію про потрібний йому товар;
- *брокеринг товарів* – фаза пошуку інформації з метою оцінки альтернативних товарів на основі критеріїв, заданих покупцем;
- *торговий брокеринг* – фаза вибору продавця, що пропонує потрібний покупцеві товар, на основі критеріїв покупця (ціна, гарантії придатності, термін доставки, репутація продавця тощо);
- *переговори* - фаза погодження умов угоди (переговори можуть мати різну тривалість і складність залежно від специфіки ринку та його суб'єктів: на деяких ринках роздрібної торгівлі ціни й інші аспекти угоди фіксуються поза переговорними процесами, тоді як на інших ринках – наприклад, ринках акцій, автомобілів – угоди щодо ціни та інших аспектів угоди встановлюють в процесі переговорів);
- *платіж і доставка продукції* – ця стадія відбувається після закінчення переговорів, якщо була досягнута угода купівлі-продажу;
- *обслуговування та оцінка* – післяпродажна стадія сервісного обслуговування продукції, аналіз процесу використання товару покупцем та рівня його задоволеності.

Таб.4.1. відображає, які фази моделі CBB функціонально підтримують різні ПА *е*-комерції.

Таблиця 4.1

Фази моделі CBB

Стадії торговельної	Personal Logic	Firefly	Bargain Finder	Jango	Kasbah	Auction Bot	Tete-a-

операції							tete
Ідентифікація потреби							
Брокеринг товарів	+	+		+			+
Торговий брокеринг			+	+	+		+
Переговори					+	+	+
Платіж і доставка							
Обслуговування та оцінка							

Агентно-орієнтовані застосунки е-комерції

Розглянемо детальніше властивості деяких ПА, що застосовуються в *е*-комерції. *PersonaLogic* (http://www.personalogic.com) дозволяє споживачам сформувати список товарів, які найкраще відповідають їх потребам. Система в межах певної ПрО відфільтровує товари, що задовольняють жорсткі і м'які обмеження, задані користувачем [175].

Firefly (http://www.firefly.com) порівнює параметри товарів, які цікавлять споживача, з тими оцінками, які надають цим товарам інші споживачі. Система знаходить споживачів, інтереси яких найближчі до інтересів користувача ПА, і після цього рекомендує користувачеві ті товари, що ці споживачі оцінили найвище [59].

BargainFinder – віртуальний ПА покупок у режимі он-лайн [16]. Покупець описує, що саме він хоче купити. BargainFinder запитує ціну товару, який цікавить його користувача, у кожному з різних комерційних Web-вузлів, та повертає користувачеві отримані результати. *Jango* [84] може розглядатися як розвинутий BargainFinder. ПА отримує запити безпосередньо від агентів покупців, а не з централізованого сайта. Як тільки будь-який продавець зі списку Excite надає відкритий каталог з цінами товарів, які він пропонує, Jango знаходить у ньому товари, які шукає один з ПА покупців, та порівнює ці ціни з пропозиціями інших продавців

Kasbah [109] – МАС *е*-комерції. Споживач створює ПА, задає йому певну стратегію і відсилає цього агента до централізованого електронного ринку ПА. ПА ринку автоматизують стадії

комерційного посередництва і переговорів як для покупців, так і продавців. Агенти продаж спілкуються з відповідним ПА, пропонуючи ціну на товар та отримуючи повідомлення "так" або "ні". У випадку відмови ПА збільшує ціну, яку він пропонує продавцю. Для цього ПА Kasbah забезпечує агентів однією з трьох стратегій збільшення пропозицій ціни на товар через певний проміжок часу: "активно", "спокійно" та "раціонально" – відповідно до лінійної, квадратичної та експоненціальної функцій.

ПА Kasbah шукають потенційних клієнтів або торговців і ведуть переговори з ними від імені їх власників. Мета кожного ПА – укласти угоду, що задовольняє набору вказаних споживачем обмежень, який містить, наприклад, бажану ціну, максимальну (або мінімальну) ціну, дату покупки тощо. Ринок Kasbah розроблений так, щоб підтримувати будь-які ПА, які спроможні спілкуватися з використанням відповідного протоколу.

AuctionBot [13] - це сервер Інтернет-аукціону Мічиганського університету. Щоб продавати свою продукцію, користувачі AuctionBot створюють власні аукціони, обираючи один з запропонованих типів, а потім конкретизуючи його параметри (клірингові умови, метод вирішення пропозиції цін, кількість дозволених торговців тощо). Клієнти і продавці можуть запропонувати ціну згідно із багатостороннім дистрибутивним протоколом переговорів аукціону. У типовому сценарії торговець може пропонувати попередню ціну після створення аукціону і дозволяє AuctionBot взаємодіяти з продавцем, який пропонує ціну згідно із протоколом аукціону і відповідними параметрами. AuctionBot забезпечує інтерфейс API застосунків для споживачів, що дозволяє їм створювати власних ПА, здатних автономно діяти на ринку AuctionBot. Споживачі мають змогу кодувати власні стратегії пропозиції ціни.

Tete-a-Tete (http://ecommerce.media.mit.edu/tete-atete) використовується як посередник між покупцями і продавцями, реалізуючи переговори між ними [218]. Tete-a-Tete забезпечує роздрібний продаж товарів. ПА цієї МАС ведуть переговори щодо умов транзакції – приміром, гарантійного обслуговування, часу доставки, варіантів позики тощо. Переговори не використовують функцій підвищення та спаду цін, як в Kasbah. ПА Tete-a-Tete підтримують послідовний стиль переговорів ПА продажу, використовують обмеження на визначення вартості товару на стадіях вибору товарів і визначення продавця.

MAGMA (Minnesota Agent Markerplace Architecture) [121] – це прототип віртуальної системи ринку. Поточна реалізація складається з сервера підтримки, написаного в Allegro Common Lisp, і набору ПА, написаних мовою Java, які функціонують в WWW. Зараз MAGMA призначена для торгівлі товарами, які легко пересилати через Інтернет.

Система MAGMA складається з таких об’єктів, як ПА продавця, сервера реклами та банку.

Ролі об’єктів MAGMA:

- ПА продавця відповідають за покупку і продаж товарів, управляють переговорами щодо цін;
- сервер реклами забезпечує послуги систематичної реклами, яка включає інформаційний пошук за категоріями;
- банк забезпечує набір основних банківських послуг, які включають перевірку рахунків, розміри дозволеного банком кредиту та електронної готівки.

У MAGMA всі ПА функціонально незалежні і спілкуються, використовуючи сокети. Для спрощення комунікацій між ПА система MAGMA використовує сервер ретрансляції, який підтримує зв'язки між сокетами і надсилає повідомлення ПА, базуючись на унікальних іменах цих ПА. Ця МАС є відкритим стандартом, дозволяючи платформам і незалежним ПА підключатися до сервера ретрансляції і управляти бізнес-процесами через інфраструктуру MAGMA. Це дозволяє MAGMA інтегруватися з іншими системами, наприклад, банківськими для проведення транзакцій.

Інша відома система *e*-комерції – *XPect*[121]. Структура XPect підтримує ПА, які можуть інкапсулювати комерційні застосунки або послуги, під’єднуватися й координуватися, використовуючи мовні засоби координації CLF для реалізації комерційних транзакцій. *CLF* (Coordination Language Facility) – це об’єктно-орієнтований інструмент розробки застосунків, який має вигляд об'єктної моделі автономних ПА та дозволяє динамічно додавати нові послуги, а також координувати доступ до багаторазових послуг. Координація в XPect відбувається у різних формах:

- переговори відбуваються, наприклад, між замовником і постачальниками, щоб конкретизувати атрибути множини об’єктів інтересу, або між банкіром і постачальником, щоб визначити найкращий метод оплати (якщо замовник делегує цю задачу банкіру);

- транзакція відбувається та обробляється банкіром, коли повністю заповнено бланк замовлень;
- сповіщення може використовуватися, аби замовник був поінформований щодо відвантаження товару в процесі доставки.

Ці дії ПА підтримуються операціями CLF на базовому рівні. CLF забезпечує засновані на правилах мови сценарії, підтримує комплексну поведінку координації ПА, що охоплює як запити, так і операції, типові вимоги *е*-комерції.

У формальній моделі, запропонованій Паубеллі та Куннінгхамом, присутня фаза автоматичних переговорів [173]. Модель використовує механізм деонтичної логіки, щоб подавати ситуації та процеси системи переговорів.

У багатьох ситуаціях покупці бажають використовувати власні персональні дані для підвищення ефективності трансакцій е-комерції (інформацію щодо сфери їхніх інтересів, переваги, місця проживання, мови взаємодії тощо), але сучасне ПЗ не завжди може підтримувати ці можливості та часто обробляє інформацію тільки в оригінальному форматі [269]. Саме ПА здатні задовольнити це бажання покупців.

Більшість сучасних застосунків електронної комерції – досить просте за функціями ПЗ. Це сервери, підключені до Інтернету, що дозволяють користувачам переглядати каталоги, що містять інформацію про товари і послуги. Ключова проблема таких систем – те, що вони концентруються лише на одному аспекті транзакції – ціні. Але часто замовники і підприємства враховують більш важливі параметри для прийняття рішень про торговельну угоду: підтримку регулярних та оптових замовників, гарантійне обслуговування, час доставки тощо.

Розвинутіші системи (як, Jango і Bargainfinder) здатні виконувати комерційне посередництво від імені користувача, а система Tete-a-Tete намагається підтримувати покупця на кількох фазах торговельної операції. Розглянемо ще кілька цікавих прикладів застосування агентних технологій *е*-комерції.

МАС *iJADE* – приклад інтелектуального середовища Java для розробки ПА, яке забезпечує інтегровану та інтелектуальну агентно-орієнтовану платформу [119].Застосування цієї МАС призводить до автоматизації процесу купівлі в режимі он-лайн. Якщо IBM Aglets та ObjectSpace Voyager Agents в основному забезпечують властивості мобільності та комунікації МАС, то iJADE забезпечує інтелектуальний рівень, підтримуючи ширшу функціональність застосунків МАС.

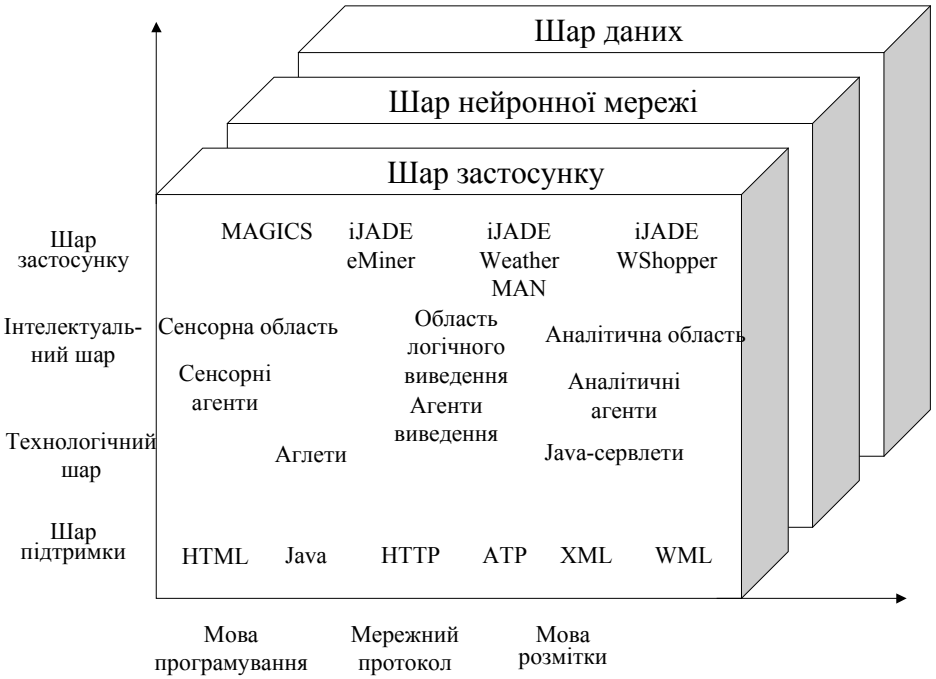


Рис.4.1. Структура системи iJADE

Один з найвідоміших застосунків iJADE (рис.4.1) в середовищі *e*-комерції – *iJADE eMiner,* який складається з двох головних модулів: 1) здобуття візуальних даних та системи візуалізації для автоматичної аутентифікації персональних даних за зразками, який базується на Fagent–моделі, що використовує техніку EGDLM [120]; 2) Web-інструмент, що базується на нечітких нейронних алгоритмах з метою пошуку та брокерингу товарів – FShopper. Аналіз Web-контенту (HTML- та XML-документів) забезпечує здобуття з нього знань. В інтелектуальній моделі ПА iJADE подано два інноваційні мережні застосунки iJADE eMiner: 1) FAgent, автоматичні візуальні дані здобуття ПА, заснованого на EGDLM (еластична динамічна графова модель зв'язку) для автоматичної ідентифікації, що базується на інваріантному людському розпізнаванні персони; 2) Fshopper – Web-орієнтований застосунок пошуку товарів, що використовує ПА купівлі на базі нейронних та нечітких алгоритмів.

IJADE має два рівні абстракції: рівень системи iJADE – модель дій та рівень даних iJADE – модель DNA. Модель DNA містить рівні даних, нейронної мережі та застосунків, а модель дій – прикладний, інтелектуальний, технологічний та підтримки.

Модель iJADE DNA забезпечує структуру маніпулювання даними, яка базується на технології нейронних мереж. Рівень даних відповідає необробленим даним. Рівень нейронної мережі забезпечує кластеризацію різних типів нейронних мереж з метою організації,

інтерпретації, аналізу та прогнозування дій, заснованих на подачі від рівня даних, які використовуються застосунками iJADE.

Особливість системи iJADE – це метод дій, який забезпечує пошарову архітектуру для виконання МАС.

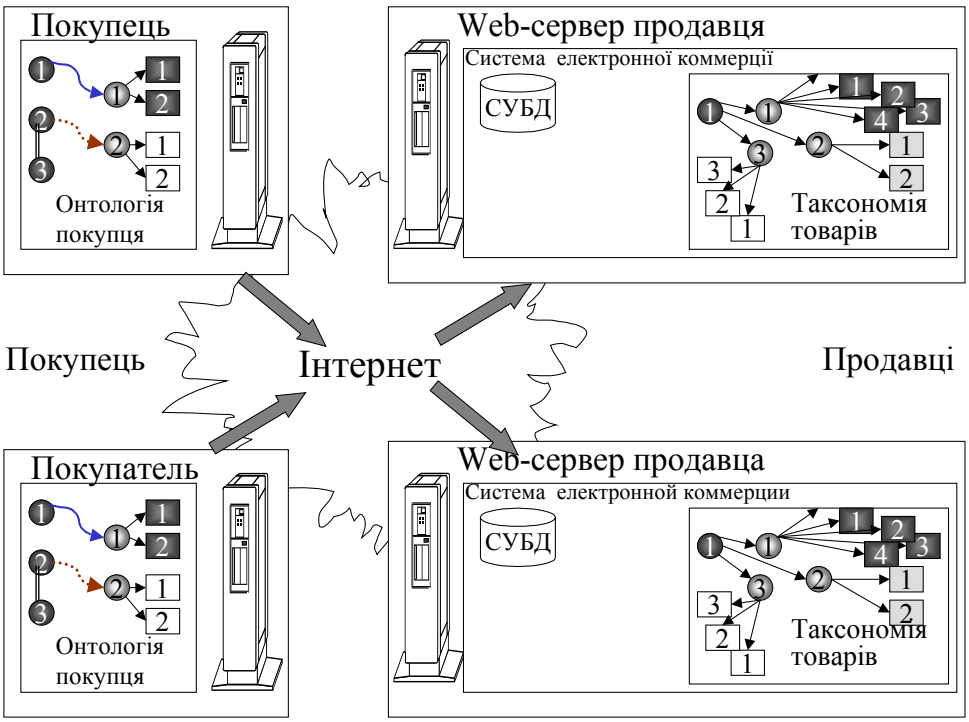
Прикладний рівень iJADE eMiner – це найвищий рівень, який складається з інтелектуальних агентно-орієнтованих застосунків. Застосунки, що здійснюються на цьому рівні, містять:

- MAGICS (Mobile Agent-based Internet Commerce System) – набір інтелектуальних агентно-орієнтованих застосунків *е*-комерції, який складається з MAGICS-покупця (ПА покупок у Інтернеті) і аукціону MAGICS (інтелектуальний ПА аукціону);
- iJADE eMiner – МАС, що містить: 1) FAgent – автоматичну систему ідентифікації, засновану на розпізнаванні особи; 2) Fshopper – ПА покупок у Інтернеті;
- iJADE WeatherMAN – інтелектуальний ПА прогнозування погоди, підтримується властивістю прогнозування нейронної мережі;
- iJADE Wshopper – інтелектуальний ПА на основі нечіткої логіки з використанням технології WAP для інтелектуальних мобільних покупок у Інтернеті.

Архітектура інтелектуальної МАС е-комерції

Якщо користувач має модель переконань, знань, намірів і цілей системи, то він може розуміти те, які дії ця система вибирає в різних ситуаціях. Це вирішує соціальну проблему агентних застосувань: люди не хочуть використовувати ПЗ з непрогнозованою поведінкою і некерованими діями, особливо у сфері комерції.

Покупці та продавці взаємодіють один з одним через середовище Інтернет, використовуючи для встановлення спільної термінології онтологічне подання знань (рис.4.2):



Специфіка торгівельних підприємств подається через онтології відповідних ПрО, з метою інтероперабельності поданих у форматі OWL (рис.4.3).

Рис.4.2. Використання онтологій для взаємодії продавців і покупців

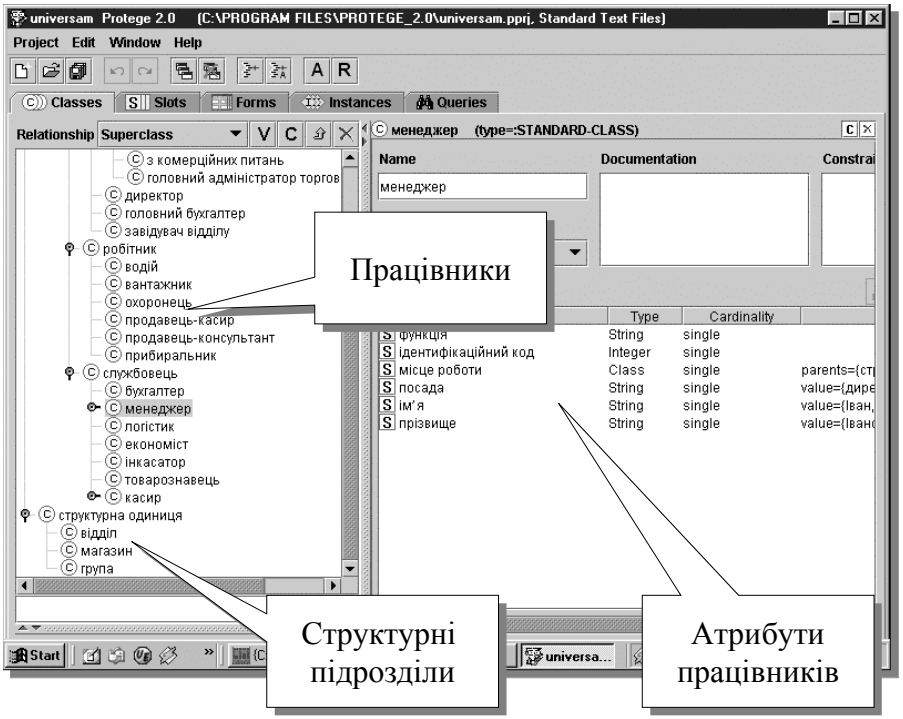


Рис.4.3. Онтологія універсаму

Основне завдання МАС *e*-комерції полягає в тому, щоб агенти продавців і покупців знайшли один одного і ратифікували таку угоду купівлі/продажу, яка б задовольнила обидві сторони. Сучасні ПА

можуть приймати рішення відносно великої кількості покупок і продажу на підставі різнорідної і розподіленої інформації.

Кожна торговельна операція подається такими фазами:

- передпродажною;
- продажною;
- післяпродажною.

Фаза, що передує продажу, містить операції ідентифікації товару покупцем, вибір товару і продавця, укладання контракту; фаза продажу – операції оплати і доставки товару; фаза після продажу – це забезпечення супроводу товару після продажу і можливості відмовлення від покупки.

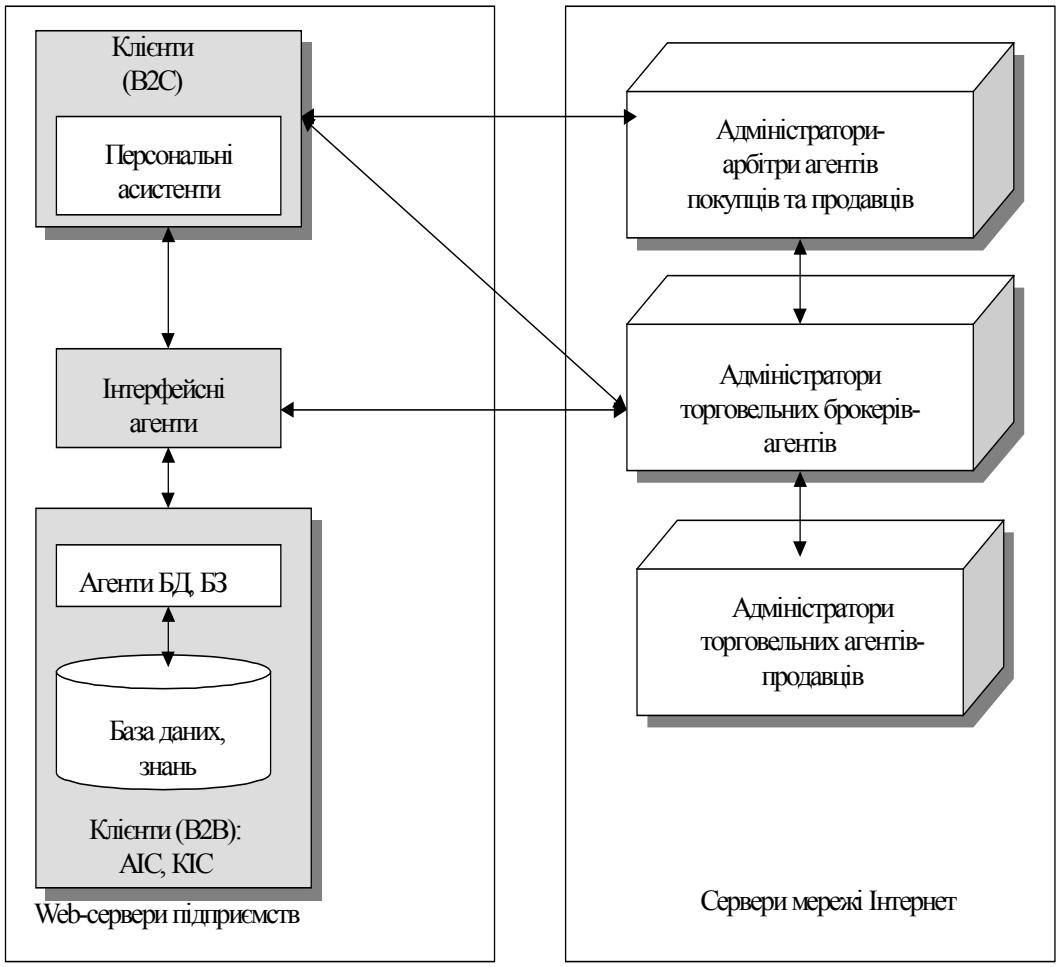


Рис.4.4. Архітектура МАС *e*-комерції

Архітектуру МАС керування замовленням .товарів та послуг у мережі Інтернет можна подати у вигляді схеми (рис.4.4) [323].

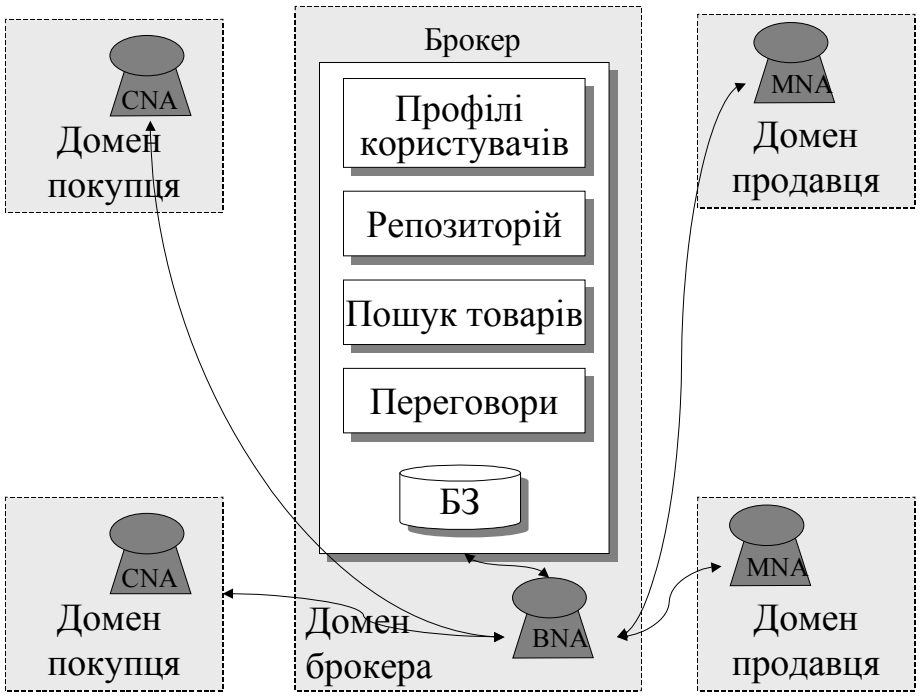


Рис.4.5. Архітектура інтелектуальної МАС *e*-комерції з використанням брокера

Віртуальний ринок допускає наявність груп покупців, продавців і брокерів, кожний з яких представлений в МАС відповідними ПА (рис.4.5). Кожен покупець, що звертається до МАС, отримує персонального ПА – CNA (Customer Negotiation Agent), якому користувач задає профіль своїх вподобань та інтересів. Цей профіль зберігається і може бути використаний при наступному звертанні користувача до МАС.

Брокер за допомогою системи оповіщень (запитів і відповідей) ізнається про бажання покупця придбати певні товари і створює для цього покупця агента переговорів з брокером – BNA (Broker Negotiation Agent), який забезпечує виконання торговельної операції.

Кожна група продавців має постійно працюючих ПА з власними профілями – MNA (Merchant Negotiation Agent), які взаємодіють із агентами BNA. Процес продажу виконується автоматично персональними ПА покупців і продавців на основі розподіленої онтології і спеціального торговельного протоколу (наприклад, FIPA-iterated-contract-net-protocol).

Вибір моделі ПА для підтримки е-бізнесу залежить від типу інформації, що обробляється, задекларованих цілей, терміну дії агента тощо. ПА можуть шукати товари, формувати групи продавців і покупців, вести переговори, автоматично описувати товари і послуги відповідно до зазначених вимог користувача.

У МАС *е*-комерції використовують:

- моделі домена *е*-комерції, що базуються на онтології виявлення відповідності між потребами покупців і пропозиціями продавців;
- персоніфікацію суб'єктів *е*-комерції;
- прогнозування потреб покупців на основі історії їхніх покупок за допомогою методів Data Mining.

Відповідно до схеми купівлі/продажу кожен ПА описується:

- комунікаціями, що управляють усіма взаємодіями;
- мовою торговельних операцій, що забезпечує спільну онтологію взаєморозуміння, можливість коректного аналізу вхідних повідомлень і відправлення вихідних повідомлень;
- протоколом спостереження допустимості вхідних і/або вихідних повідомлень;
- модулем прийняття рішень і стратегії/тактики поведінки агента з приводу індивідуалізації інформаційного наповнення.

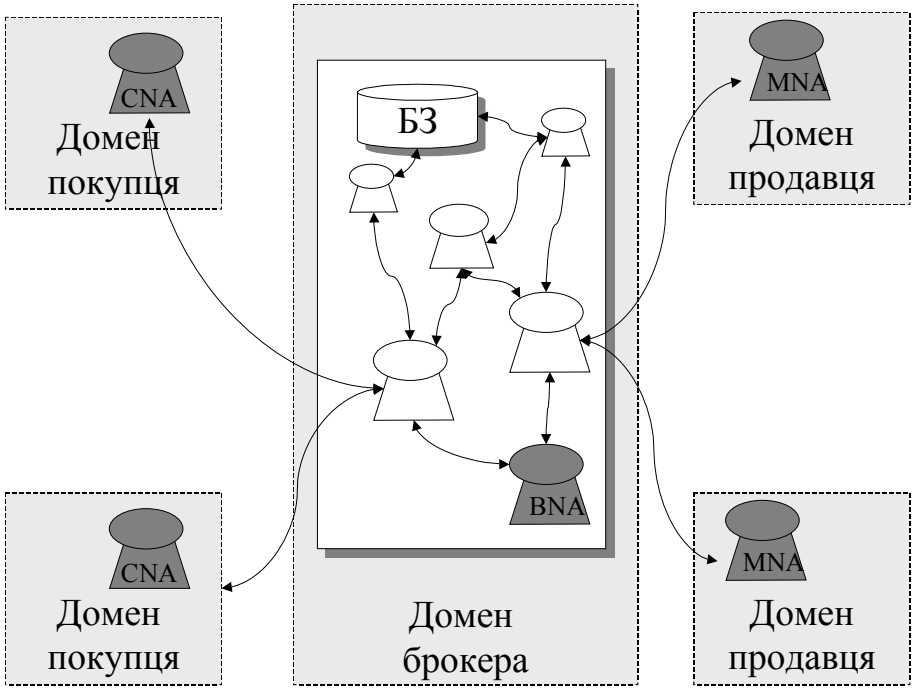


Рис.4.6. Архітектура інтелектуальної МАС *е*-комерції з використанням допоміжних ПА

Архітектура МАС може бути розширена агентами-посередниками, які виконують службові функції (обробку БД і БЗ, підтримку політики безпеки, збір інформації, аналіз онтологій доменів тощо).

Для гнучкості і масштабування ПЗ в МАС використано різних спеціалізованих допоміжних ПА (рис.4.6):

- реєстратора для роботи з БД;
- диспетчера пропозицій продавців;
- диспетчера запитів покупців;
- інтегратора, що порівнює запити покупців із пропозиціями продавців;
- консультанта, що перетворює запит користувача на основі онтології домена;
- агента безпеки, що захищає конфіденційну інформацію щодо процесу *е*-комерції.

Переваги такої МАС *е*-комерції:

- система працює на різних платформах (з використанням мови програмування Java і БД MySQL, які є доступними);
- агенти-посередники забезпечують гнучкість і масштабування цього ПЗ;
- мови комунікації агента (KQML і KIF) забезпечують взаємодію з агентами інших розробників;
- мови (OWL) і інструменти побудов онтологій (Protégé, OntoEdit) використовуються для розробки онтології домену;
- для пошуку релевантних термінів використовуються оригінальні алгоритми аналізу онтології ;
- прогнозування потреб покупців здійснюється на основі історії їхніх покупок.

ПА брокерингу використовують знання і переконання, що мають ПА покупців і продавців, але їхні цілі складніші й орієнтуються на виконання максимальної кількості транзакцій *е*-комерції. Важлива умова їхньої роботи – не віддавати переваг деяким продавцям за рахунок інтересів покупців.

Якщо модель цього агента містить такі обмеження, то користувачі можуть переконатися в їх неупередженості.

У [269, 270] розглянуто МАС *е*-комерції і її реалізацію на Java, що містить агенти продавців (3 типи), агенти покупців (3 типи) і один загальний помічник агента (рис.4.7).

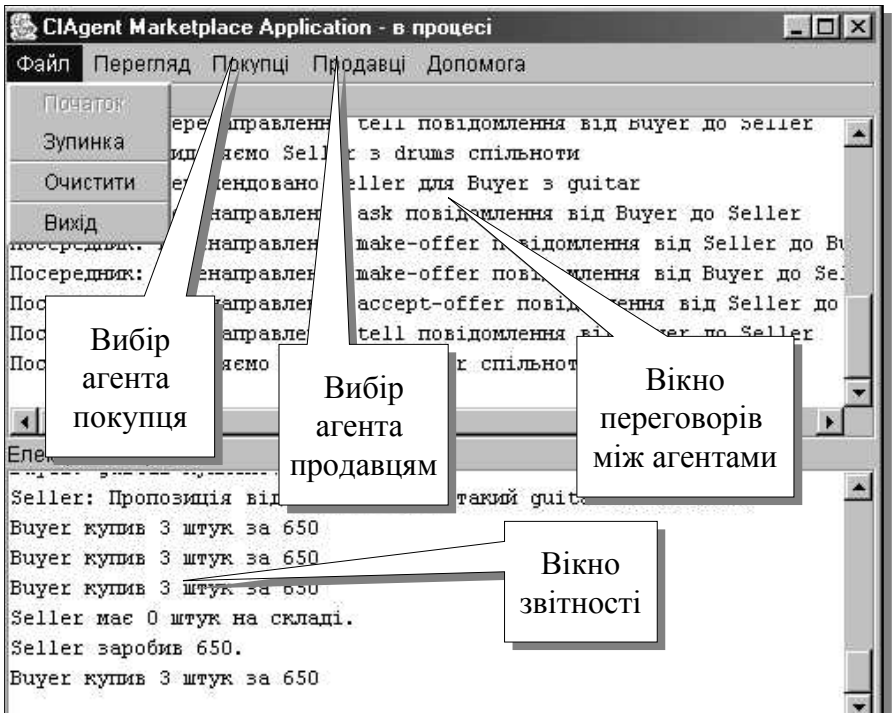


Рис.4.7. МАС *e*-комерції

ПА-помічник надає інтерфейс між продавцями і покупцями.

Ця МАС містить такі модулі:

- Offer – визначення ціни товарів на пропозиціях;
- Basic Negotiation – менеджмент на основі переговорів;
- AboutDialog – менеджмент діалогу з користувачем;
- Marketplace Frame – координація і керування МАС;
- BaySel Message – інтерфейс взаємодії з користувачем;
- FacilitatorAgent – агент-посередник;
- BayerAgent , Better BuyerAgents, Best BayerAgent – агенти покупця різного рівня складності;
- SellerAgents, Better SellerAgents, Best SellerAgents – агенти продавця різного рівня складності.

Деякі агенти мають просту архітектуру, опис якої зрозумілий для кожного користувача – продавця або споживача. Якщо ж вони мають потребу в ефективнішій обробці транзакцій *e*-комерції, то застосовують складніші агенти.

У цій МАС використовуються три рівні складності архітектури ПА продавця або покупця. МАС функціонує в корпоративній мережі. Агенти МАС використовують мову KQML для взаємодії комунікацій.

Розроблена система може використовуватися для корпоративної *e*-комерції відповідно до схем B2B, B2C і C2C.

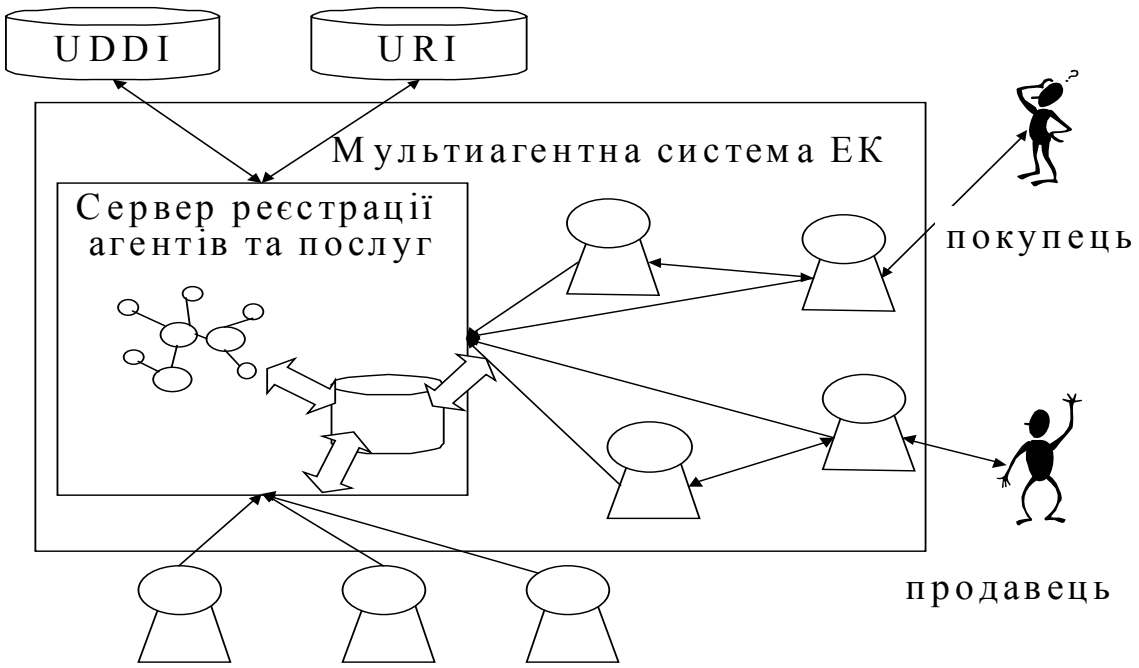


Рис.4.8. Мультиагентна система *е*-комерції

Доцільно включати до складу МАС *е*-комерції додатковий компонент – *сервер реєстрації* ПА та послуг, які вони надають [324]. Агенти, що розроблені в процесі створення МАС *е*-комерції, – "внутрішні" агенти – реєструються розробниками на цьому сервері.

Ці агенти забезпечують мінімальний набір послуг для здійснення *е*-комерції. Агенти, що звернулися до сервера реєстрації ззовні ("зовнішні" агенти), реєструються адміністратором сервера на основі тієї інформації, яку вони про себе повідомляють (рис.4.8).

Крім того, експорт сервісів з UDDI (Universal Description, Discovery and Integration) забезпечує доступ до послуг, які надають ПА, що не зареєструвалися на сервері (пошук необхідних послуг здійснюється при надходженні запиту на відповідну послугу, якщо така послуга не зареєстрована на сервері).

У БД сервера реєстрації зберігається така інформація про кожного ПА: його ідентифікатор, перелік послуг, які агент надає, та параметри виклику цих послуг.

ПА е-бізнесу дозволяють реалізовувати бізнес-логіку обробки IP. ПА отримує від споживача запит, що містить опис продукції, яку споживач хоче купити. Після цього ПА шукає серед Web-ресурсів інформацію про цю продукцію та постачальників, які її пропонують. Агент прагне надати споживачеві тільки релевантні посилання. У процесі взаємодії ПА зі споживачем параметри запиту (ціна, час

доставки, гарантійні зобов'язання тощо) можуть бути скориговані залежно від пропозицій постачальників. Агентно-орієнтована технологія надає механізм для обробки відновлень за змістом та структурою бізнес-трансакцій.

Пропонується використовувати для опису товарів, які хоче придбати покупець, простори імен, що описують предметну область продавця. Агент-брокер обробляє опис товару, який бажає придбати покупець, та знаходить продавців, які пропонують товари, що відповідають запиту користувача, з однаковими умовами. Потім він для того, щоб визначити того з них, з ким буде укладена угода, за допомогою ІПС Інтернету знаходить Web-сайти продавців і визначає простори імен, що використовуються для XML-описів своїх ресурсів. Потім він порівнює простори імен покупця і продавця, щоб визначити, хто з продавців ближче за сферою спеціалізації до потреб покупця. На рис.4.9. показано процес обробки запиту споживача агентом-брокером [324,325].

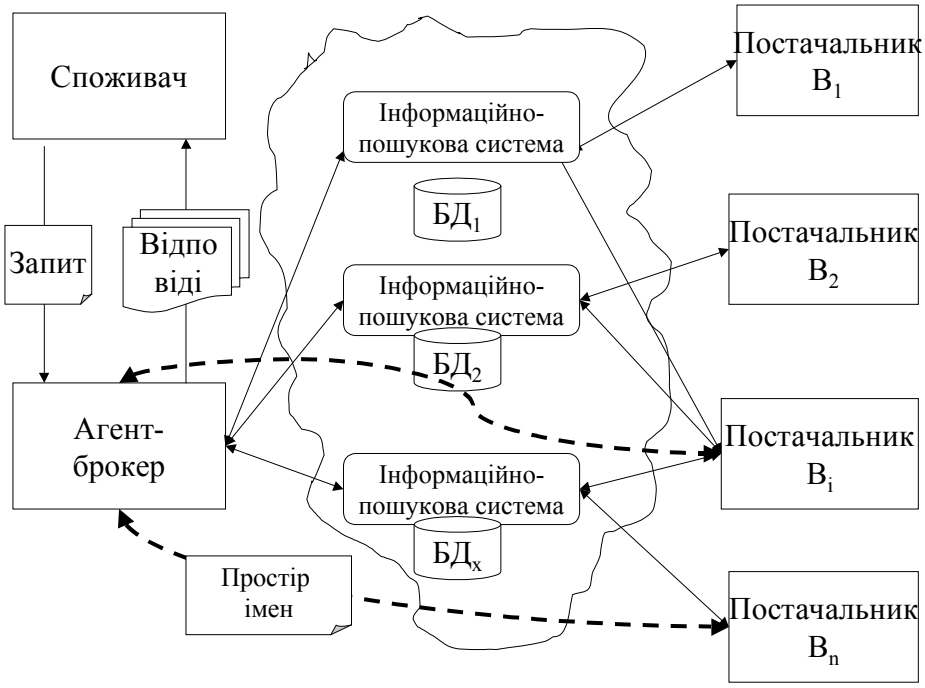


Рис.4.9. Процес обробки запиту споживача агентом-брокером.

Аналізатори XML-документів дозволяють одержати інформацію, що відноситься до простору імен: базове ім'я вузла, префікс простору імен, ідентифікатор URI простору імен, що відповідає префіксу простору імен вузла і т.д. Наприклад, у визначенні простору імен *xmlns:catalog="http://www.e-torg_it.com/my_name_catalog.dtd" xmlns* – зарезервоване ключове слово для декларації простору імен, *catalog* – префікс простору імен, а

"*http://www.e-torg_it.com/my_name_catalog.dtd*" – URI цього простору імен. Проаналізувавши опис простору імен, можна сформувати множини термінів ПрО з урахуванням їх ієрархії (тобто одержати набір множин – «терміни 1-го рівня», «терміни 2-го рівня», …, «терміни і-го рівня»). Порівнюючи близькість таких наборів множин (з урахуванням ієрархії), можна кількісно оцінити релевантність запиту споживача ресурсам, що пропонує постачальник.

$$R(A,B)=\sum_i\sum_j \frac{|a_i \cap b_j|}{|a_i \cup b_j|} * C_{i_j} ,$$

де $|x|$ – кількість елементів у множині x, a_i - множина термінів і-го рівня ієрархії споживача A, b_j - множина термінів j-го рівня ієрархії постачальника B, C_{i_j} – коефіцієнт оцінки близькості термінів різних рівнів ієрархії, $0 \le C_{i_j} \le 1$, причому $C_{i_j} = 1$ для $i = j$.

Після цього агент рекомендує споживачу продукцію постачальників, що пропонують найбільш релевантні ресурси.

4.3. Програмні агенти логістики

Один з напрямків і-економіки пов'язаний зі створенням холонічних систем, у яких кожен підрозділ компанії виявляється подібним невеликій автономно діючій фірмі, а всі разом вони утворять холонічну мережу, здатну динамічно реконфигуруватися і гнучко адаптуватися до запитів ринку. При цьому кожному холону необхідно приймати самостійні рішення, відповідаючи на різноманітні питання типу "*Чи можемо ми виконати замовлення X і якщо так, то в які терміни*?", "*Хто має брати участь у виконанні замовлення X?*".

Наукова розробка [262] присвячена створенню холонічних компаній з мережною організацією. Однією з важливих проблем створення таких компаній виявляється проблема прийняття рішень у ході різко зростаючою числа переговорів у мережі. Для рішення цієї проблеми пропонується мультиагентна система, що може бути використана для моделювання процесів переговорів і погодженого прийняття рішень. Розроблений підхід використовується для побудови холонічної мережі торгової компанії.

Для досягнення взаємовигідних домовленостей потрібні інтенсивні переговори, спрямовані на узгодження прийнятих рішень і досягнення балансу інтересів. Крім того, для прийняття оптимальних рішень потрібно перебирати багато варіантів таких домовленостей і

знайти оптимальний варіант рішення задачі. На практиці трудомісткість таких переговорних процесів виявляється дуже висока і потребує від менеджерів великих зусиль та багато часу. Тому для моделювання переговорів доцільно застосовувати МАС, що дозволяє істотно прискорити процеси прийняття рішень на всіх ділянках мережі холонічного підприємства, агенти яких можуть представляти у переговорах інтереси та можливості окремих холонів і здійснювати процес переговорів значно швидше порівняно з тим, як це роблять люди.

У багатьох наукових працях [316, 202] розглядається застосування мультиагентного підходу для рішення задач логістики. Як основні задачі розглядаються задачі конструювання і моделювання мереж виробництва і збуту продукції.

Логістика – процес прийняття рішень у сфері нормування і розподілу ресурсів, тобто комплекс заходів для задоволення попиту на ресурси в заданому місці й у заданий час

Типові задачі логістики систем виробництва і збуту продукції:

- конструювання логістичної системи виробництва і збуту продукції з метою забезпечення певних заданих характеристик, в процесі якого зважується задача географічного розташування цехів і складів, визначається пропускна здатність доріг, обсяги складів, аналізуються характеристик парку транспортних засобів тощо;
- моделювання створеної логістичної системи з метою дослідження її основних характеристик, у ході якого можливо визначення вартості і термінів виконання кожного замовлення, побудова оптимальних шляхів і планів виробництва і постачань, визначення «вузьких» місць системи (нестача складських приміщень, низька швидкість перевезень і т.п.), відпрацьовування аварійних ситуацій (таких, наприклад, як утрата частин) тощо.

Основою для розгортання мережі виробництва і збуту продукції є структура міст і шляхів (шосейних, повітряних, водних).

Пропонується використовувати інтелектуальних ПА, які представляють вироблену продукцію і її окремі частин [263]. Шляхом переговорів між такими ПА забезпечуються висока гнучкість і ефективність, а також надійність постачань у задані терміни. Як приклади застосування МАС демонструються діючі макети-прототипи для моделювання виробництва і продажу автомобілів [253], а також паперового виробництва і його продажу .

В архітектурі МАС основну частину становить предметно-незалежне мультиагентне ядро, у складі якого виділяються такі базові компоненти (рис.4.10):

- служба прямого доступу забезпечує безпосередній доступ візуальної компоненти до атрибутів агентів за допомогою повідомлень;
- служба повідомлень відповідає за передачу повідомлень між ПА, а також за комунікації ПА з додатковими системами ядра;
- бібліотека класів агентів – частина БЗ, що містить інформацію щодо типів ПА. Для підвищення гнучкості системи, інформація в цьому довіднику може бути доповнена з зовнішніх розширень через інтерфейс EM API (розширення ядра);
- співтовариство ПА – блок, що забезпечує життєдіяльність ПА, а також функції завантаження ПА і їхніх властивостей і оптимізацію роботи з ресурсами;
- онтологія – предметна БЗ, що містить конкретні знання про предмет (логістику), подані у вигляді семантичної мережі.

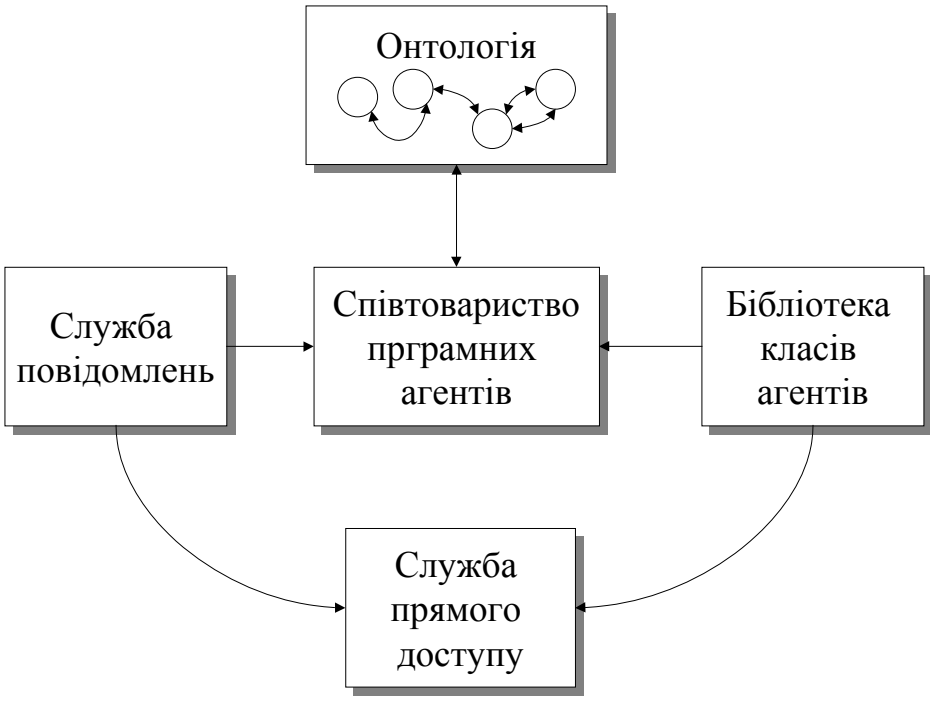


Рис. 4.10. Архітектура ядра МАС логістики

Коли надходить замовлення на деталь, .відразу створюється ПА замовлення для проведення переговорів про прийняття його у виробництво. ПА замовлення знаходить вже існуючу готову деталь на одному зі складів. Він пропонує певну ціну за дану деталь, а ПА деталі, у свою чергу, може прийняти цю пропозицію або відмовитися від неї, залежно від запропонованої ціни. Якщо ПА деталі приймає

пропозицію, тоді деталь резервується. Припустимо, що далі в систему надходить інше замовлення, і інший ПА замовлення намагається зарезервувати для себе ту ж деталь, оскільки інший потрібної не знаходить. Розуміючи, що деталь уже зайнята, ПА замовлення пропонує скасувати бронювання за певну компенсацію. У результаті ПА деталі знову звертається до першого ПА замовлення з пропозицією прийняти компенсацію. Перший ПА замовлення може погодитися і прийняти компенсацію, якщо тільки це не вплине на терміни його виконання, або дата постачання може бути перенесена на пізніший термін. Якщо перший ПА замовлення згоден, тоді деталь бронюється заново в наступному замовленні.

Так, у ході переговорів агенти можуть реалізовувати різні стратегії. Наприклад, «обережний» ПА замовлення деякої деталі послідовно обходить весь ринок пропозицій виробників, торгується з ними й у результаті знаходить найкращу пропозицію (яке, потім, до моменту ухвалення рішення вже може бути зняте), а «рішучий» – негайно бронює першу ж пропозицію, якщо вона відповідає його вимогам. Вступаючи в переговори, агенти можуть як конкурувати між собою, так і кооперуватися для досягнення заданої мети.

Крім набору агентів замовлень і готових частин, які постійно шукають зустрічі, у системі існує і ряд інших, більш традиційних додаткових агентів: агенти виробника, складу, складального виробництва (конвеєра) тощо. При цьому агент складу стягує із замовлень плату за збереження частин і постійно дозамовляє частини, постійно оцінюючи співвідношення між мінливим попитом та пропозиціями. Агент конвейєра керує критеріями складання і при необхідності може переключати переговори замовлень з одержання максимального прибутку, наприклад, на резервування часу через ремонт складальної лінії.

Предметна орієнтація МАС досягається шляхом створення додаткових класів ПА у системі Delphi і впровадження їх через EM API, а так само через розширення предметної БЗ.

Крім перерахованих вище блоків, система має чотири типи інтерфейсів для взаємодії з зовнішніми компонентами:

- DBI API – інтерфейс взаємодії з модулем доступу до БД і БЗ;
- UI API – інтерфейс системи спостереження/редагування (кожен проект може використовувати як уніфіковану оболонку для роботи із системою, так і свій власний блок, наприклад для Web-подання);
- EM API – інтерфейс розширень ядра, що дозволяє динамічно вносити в систему власні типи ПА без модифікації самого ядра;

- CI API – інтерфейс взаємодії між різними ядрами.

4.4. Програмні агенти електронних ринків

З появою *е*-комерції ринок як місце проведення операцій купівлі-продажу зазнав кардинальних змін. Компанії дістали можливість продажу та купівлі товарів через нові електронні ринки та електронні торговельні майданчики [329].

Електронний ринок (*е*-ринок) – віртуальний простір взаємодії клієнтів та постачальників для ведення *е*-комерції, призначений для придбання та реалізації товарів або послуг різної галузевої належності за допомогою телекомунікацій.

Портал е-ринку є точкою входу до глобальної мережі всіх учасників процесу е-бізнесу, а також місцем для розміщення електронних каталогів товарів, сервісів, керування транзакціями, логістичними процесами, платежами тощо.

Переваги використання е-ринків можна поділити на три групи:

- продавці одержують можливість залучення більшої кількості покупців, накопичення актуальної інформації;
- виникає сфера діяльності для посередників, які організують збирання та аналіз інформації, процеси замовлення та оплати продукції, інтеграцію ПЗ, консультаційні послуги;
- покупці одержують можливість вільного порівняння умов угод і цін, чим ініціюють активнішу конкуренцію між продавцями.

Для подання *е*-ринків на сьогодні широко використовують агентні технології. ПА е-ринку виконують такі функції:

- посередництво між гетерогенними Web-вузлами;
- моніторинг контенту і сповіщення клієнтів;
- фільтрацію та порівняння інформації;
- надання спеціалізованих послуг відповідно до потреб клієнтів;
- допомога клієнтам у прийнятті рішень;
- діяльність від імені клієнтів у пошуках потрібного, серверний моніторинг, переговори, пропозиція цін, аукціон, здійснення транзакцій, перевезення вантажів, додаткова підтримка.

На сьогодні інтенсивно створюються ряд віртуальних організацій, метою яких є забезпечення взаємодії різних груп користувачів на *е*-ринку мережі Інтернет. Прикладом слугує система KRAFT [181], що забезпечує:

- ідентифікацію потреб, яка полягає у визначенні послуг або товарів;
- вибір партнера або групи бізнес-партнерів, які разом організовуються для сумісної підприємницької діяльності;

- керування транзакціями, якими через послуги або товари забезпечуються споживачі;
- ліквідацію – розпускання групи партнерів, відбувається при цьому завершальна проплата або транзакція закриття (рис.4.11).

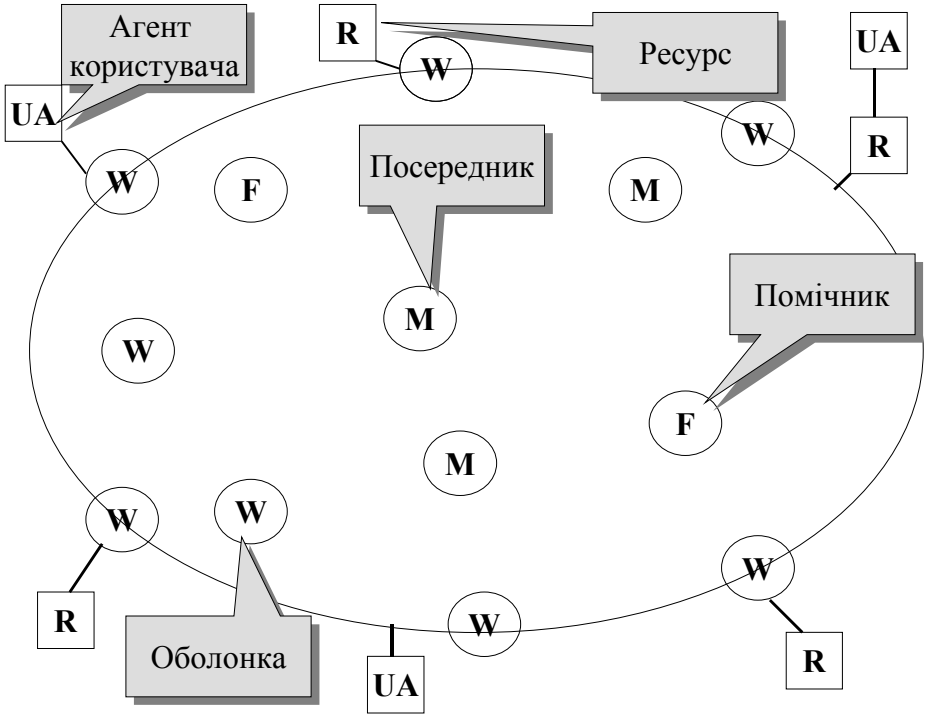


Рис.4.11. Архітектура KRAFT

Розглянемо як приклад агентно-орієнтовану інфраструктуру електронного ринку *SICS MarketSpace* [69]. Основні компоненти SICS MarketSpace – інформаційна модель опису інтересів споживача, об'єктів, контрактів тощо та модель взаємодії, яка визначає базовий словник для пошуку, переговорів і укладення угод.

Інформаційна модель базується на структурованих документах, які являють собою контракти і подання множин інтересів щодо контрактів. Імена елементів контрактів подаються через URL-адреси, що посилаються на їх визначення типу. Типи елементів (концепти) можуть бути організовані в успадковані ієрархії. Мова "інтересів" пропонує здатність надавати альтернативи для будь-яких елементів, узагальнювати успадковані ієрархії, і надавати діапазон значень. На разі розроблено мову споживачів - формат ринку інтересів MIF (Market Interest Format), що використовується з метою подання кодів ринків та товарів. W3C RDF [193] є достатнім засобом подання цієї мови програмування.

Модель взаємодії – це асинхронний зв'язок повідомлень у мовних актах, що базуються на мові взаємодії ринку MIL (Market Interaction Language) засобами KQML і FIPA ACL [76], та синтаксисі

Common Lisp. Порівняно з іншими мовами, її набір типів повідомлень – невеликий: ASK, TELL, NEGOTIATE, OFFER, ACCEPT, and REJECT. Пересилання повідомлень типів OFFER, ACCEPT, та DECLINE означає створення обов’язкових до виконання зобов'язань, і вимагає підтримки цифрових електронних підписів або інших механізмів ідентифікації.

Платформа SICS AgentBase для ПА – це платформа ПЗ для ПА, розроблених в Пролозі SICStus, SICS AgentBase для SICStus [69], що містить бібліотеки, що підтримують основні функції ПА, стандартні формати Інтернету і протоколи, KQML/KIF, MIL/MIF. Певні ПА можуть співпрацювати з процесами операційної системи. Повідомлення, як правило, відправляють та отримують асинхронним способом через TCP/IP.

Адреса ПА - це URL типу map://domain:port/agentname, де MAP (Market Agent Protocol) підтримує протокол ринкового ПА. Повідомлення ПА можуть пересилатися через SMTP, FTP, і HTTP. Іншим способом зв’язків між ПА MIL є можливість надсилати та отримувати запити HTTP. Web-адреса ПА за замовчуванням - http://domain: port/agentname.

Shopbot – це сервіс, який автоматизує покупки в мережному електронному середовищі. Наприклад, Web-орієнтований сервіс Jango [84] забезпечує простий інтерфейс для пошуку і покупки в електронних магазинах. Його оператори використовують інструменти, які автоматизують створення інтерфейсів до Web-орієнтованих магазинів. Оскільки як інформація, так і взаємодія стандартизовані, в SICS MarketSpace, створення Jango-like shopbot – є нескладним завданням.

Для персоніфікації послуг Fire-Fly [77] забезпечує механізм автоматичного сумісного використання профілів, що базується на відкритому стандарті профілів OPS (Open Profiling Standard) [155]. У SICS MarketSpace споживач тільки забезпечує інформацію для певної мети і слугує стимулом для своєчасної підтримки цієї інформації. Взаємодія між ПА здійснюється на CBL (Common Business Language).

Головна мета MarketSpace – забезпечити вищий ступінь автоматизації ринку Інтернету. У будь-який час споживач може делегувати завдання ПА в режимі off-line. Тоді споживач може бути сповіщений про події через електронну пошту або GSM/SMS.

Набір об'єктно-орієнтованих платформ запропоновано для створення комерційних застосунків, що підтримують Web. Прикладом є система *CommerceNet eCo* – платформа, сумісна зі всіма

головними платформами торгівлі Інтернету. Метою проекту eCo CommerceNet була розробка специфікації для взаємодії систем *е*-комерції, у тому числі – ПА. Каркас eCo складається з двох частин: специфікації архітектури і рекомендацій щодо семантики, які пов'язані зі створенням бізнес-документів у форматі DTD і DTD Schemas для забезпечення їхньої інтероперабельності з іншими бізнес-процесами. Системи клієнтів та провайдерів послуг використовують той рівень абстракції, що відповідає їх конкретним потребам. Архітектура eCo (рис.4.12) складається з семи рівнів – від абстрактного опису до рівня індивідуальних протоколів бізнес-документів.

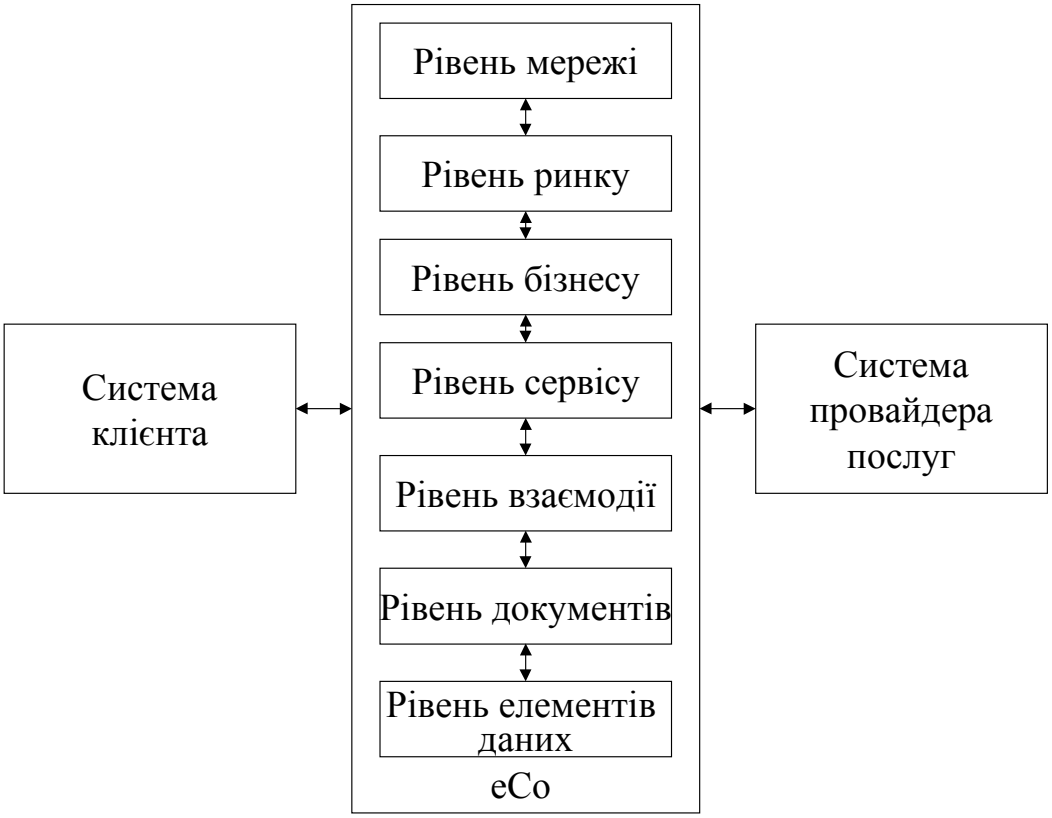


Рис. 4.12. Архітектура eCo

Теварі та Майес [126] розробили архітектуру посередників, засновану на ПА *MARI* (Multi-Attribute Resource Intermediary), для специфікації, оцінювання та посередництва гетерогенних ресурсів е-ринків. Це дозволяє забезпечити як покупцям, так і продавцям певні переваги і задавати властивості продукції в запитах. MARI надає алгоритмічну підтримку збору призначених для користувача сервісних програм, оцінюючи потенційних партнерів угод, і оптимальну відповідність інтересів клієнтів і торговців.

Сучасною тенденцією електронного ринку є перехід від торгівлі за принципом "один покупець - багато продавців" до принципу "багато покупців - багато продавців". Елементи віртуального ринку, розподілені по мережах, не прив'язані до традиційної моделі торгівлі, що забезпечувала угоду тільки одного покупця з кількома його постачальниками.

Нові моделі дозволяють зв'язати багато покупців з багатьма постачальниками, створюючи певний глобальний ланцюжок постачань в одному величезному віртуальному супермаркеті.

При цьому супровід усіх каталогів, керування всією необхідною торговою інфраструктурою, оформлення супровідної документації, доставка продукції та інші проблеми, пов'язані зі здійсненням угод, можуть централізовано підтримуватися деякою "третьою стороною", яка вирішує всі нагальні проблеми покупців.

4.5. Програмні агенти дистанційної освіти

Розвиток інформаційного суспільства забезпечує швидкий доступ до ІР по всьому світу. ІТ значно змінюють зміст і практику освіти. Зараз значна частина матеріалів, що використовуються для навчального процесу, подається в електронній формі. Використання телекомунікацій, мультимедійних навчальних ІР, Інтернет-технологій, застосунків ШІ є потенціалом для суттєвих удосконалень в освіті [78, 134]. Сучасний стан науки й освіти характеризується підвищеними вимогами до якості підготовки фахівців і означає постійний пошук нових методів і засобів підвищення ефективності освітнього процесу. Системи дистанційного освіти забезпечують адаптацію процесу навчання до індивідуальних характеристик тих, кого навчають, звільняють викладачів від трудомістких і повторюваних операцій щодо подання навчальної інформації і контролю знань, сприяють розробці об'єктивних методів контролю знань і полегшують накопичення навчально-методичного досвіду.

Розвиток ІТ, пов’язаний з використанням агентних технологій, визначає нові перспективи як в дистанційній освіті *(remote education)*, так і електронному навчанні *(e-learning)* .

Дистанційна освіта – це форма одержання освіти, що базується на комп'ютерних і телекомунікаційних технологіях. Основна її перевага – гнучкість, тобто слухачі можуть самостійно обирати час занять і визначати їх інтенсивність, знаходячись при цьому у постійному контакті з викладачем (тьютором).

Ця форма освіти здійснюється в процесі дистанційного навчання. *Дистанційне навчання* – організований за певними темами та дисциплінами навчальний процес, який передбачає активний обмін інформацією між слухачами і викладачем і застосовує сучасні засоби ІТ. Це індивідуалізований процес передання і засвоєння знань, умінь, навичок і способів пізнавальної діяльності людини, який відбувається за опосередкованої взаємодії територіально віддалених учасників навчання у спеціалізованому середовищі, створеному на основі сучасних психолого-педагогічних та інформаційно-комунікаційних технологій, у тому числі агентних. Персоніфікація в системах дистанційного навчання забезпечує активну стратегію навчання, яка дозволяє студентові керувати контекстом, темпом і межами навчання.

E-learning – навчання, яке базується на використанні обчислювальної техніки та відповідного ПЗ. Цей напрямок навчання швидко розвивається. Сьогодні значна частка студентів надає переваги електронним формам навчання. Приміром, 61% студентів в Університеті Helmut Schmidt, де використовується система ILIAS (www.ilias.uni-koeln.de) обирають саме електронне навчання.

Дві групи автоматизованих систем освіти найчастіше використовуються у мережі Інтернету: *адаптивні гіпермедіа,* АН (Adaptive Hypermedia) та *інтелектуальні системи навчання,* ITS (Intelligent Tutoring Systems). Системи АН представлені нелінійною структурою освітніх матеріалів і забезпечують користувачеві легку навігацію через гіперпосилання та зручну форму подання матеріалу. *Системи керування навчанням* LMS (learning management systems) – це інтегровані системи, які підтримують різноманітні потреби тьюторів і студентів. LMS забезпечують тьютору можливість компоновки своїх курсів з іншими навчальними ресурсами. Застосування стандартів забезпечує багаторазове використання навчальних структурних одиниць з різними цілями.

Основний принцип розробки архітектури систем дистанційного навчання полягає в уніфікації рішень. Уніфікація технологічних і організаційних рішень базується на еталонній моделі і профілях – наборах стандартів на інтерфейси різного типу.

Комітет стандартизації технологій навчання LTSC (P1484 IEEE Learning Technology Standards Committee – http://ltsc.ieee.org/) Міжнародного інституту IEEE розробляє стандарти, які забезпечують інтеграцію систем дистанційного навчання різних навчальних закладів у єдиний інформаційний простір.

Стандартизації підлягають елементи та процеси освітніх систем, при цьому особливого значення набуває подання навчального матеріалу в електронному вигляді. Для автоматизації процесу дистанційного навчання міжнародними організаціями розроблено стандарт метаданих навчальних матеріалів IEEE 1484.12.1 "Метадані навчальних матеріалів" [107]. Цей стандарт містить концептуальну модель навчальних ІР, що складається з таких категорій:

- *загальної* (General) – інформації, що описує навчальні матеріали в цілому: ідентифікатор, мову викладання, короткий текстовий опис, ключові слова, область опису, структуру, рівень агрегації;
- *життєвого циклу* (Life Cycle) – особливостей поточного стану навчальних матеріалів: версії, статусу, інформації про розробників;
- *мета-метаданих* (Meta-Metadata) – інформації про метадані навчальних матеріалів: про розробників метаданих, схему подання, мову метаданих;
- *технічної* (Technical) – технічних параметрів і характеристик навчальних матеріалів: формату, розміру, місця розташування, технічних вимог, необхідного устаткування та ПЗ, тривалості вивчення тощо;
- *освітньої* (Educational) – освітніх і педагогічних характеристик навчальних матеріалів: способи вивчення, типи навчальних матеріалів, рівні інтерактивності, типи користувачів (школяр, студент, викладач тощо), вікові категорії користувача, ступінь складності, час вивчення, описи, мови викладення;
- *правової* (Rights) – авторських прав на навчальний матеріал, а також їх умов використання;
- *відношень* (Relation) – можливостей, що визначають взаємозв'язки між навчальними матеріалами: видів зв'язків, ресурсів;
- *анотації* (Annotation) – коментарів щодо використання матеріалів;
- *класифікаційної* (Classification) – описи навчальних матеріалів відносно специфічних класифікаційних систем: цілей класифікації, інформації про таксономії, описи, ключові слова.

Для формалізації структур, що забезпечують функціонування систем дистанційного навчання Комітетом IEEE LTSC розроблено архітектуру LTSA (Learning Technology Systems Architecture).

LTSA забезпечує уніфікований підхід до моделювання і забезпечення доступу до однотипних ІР, що використовуються в різних освітніх технологічних системах. На основі цієї архітектури можна реалізувати основні функції освітніх технологічних систем, що мають такі властивості, як розширюваність функцій прикладного ПЗ,

переносність програм і даних між різними програмно-апаратними платформами, дружність інтерфейсу. На сьогодні в системі відкритої освіти використовують такі стандарти:

- IEEE 1484.1 Архітектура і Еталонна модель.
- IEEE 1484.3 Глосарій.
- IEEE 1484.2 Модель слухача.
- IEEE 1484.13 Ідентифікатори слухача.
- IEEE 1484.20 Визначення компетенції.
- IEEE 1484.10 Обмін даними при комп'ютерному навчанні.
- IEEE 1484.6 Впорядковування контенту.
- IEEE 1484.17 Упакування контенту.
- IEEE 1484.12 Метадані навчальних об'єктів.
- IEEE 1484.9 Локалізація.
- IEEE 1484.14 Семантика і зв'язування обмінів інформацією.
- IEEE 1484.15 Протоколи обміну даними.
- IEEE 1484.11 Комп'ютеризоване навчання.
- IEEE 1484.18 Профілі платформи.
- IEEE 1484.7 Взаємодія агентів і інструментів.

Реалізація функцій навчального процесу в системі відкритої освіти припускає наявність таких функціональних складових:

- підготовка дистанційних навчальних курсів і посібників;
- керування навчальним контентом;
- забезпечення зв'язку електронної бібліотеки з навчальним процесом, постачання IP Інтернету;
- адміністрування навчального процесу;
- планування процесу навчання, контроль і оцінка ступеня засвоєння індивідуальних знань осіб, які навчаються;
- комунікації в процесі навчання і адміністрування.

ПА можуть використовуватися для реалізації функцій усіх цих складових, але найбільш широко вони застосовуються для забезпечення комунікацій між об'єктами системи дистанційного навчання та ресурсами, що використовуються у цьому процесі.

Еталонна модель освітньої технологічної системи може включати такі системні компоненти:

- об'єкт навчання;
- педагог;
- оцінювання;
- постачання навчальних курсів;
- навчальні ресурси;
- успішність.

Організація роботи освітньої технологічної системи на базі еталонної моделі охоплює такі функціональні складові:

- процеси (навчання, оцінка знань, викладання, постачання);
- БД (навчальні ресурси, успішність, слухачі);
- потоки даних (інформація про переваги в навчанні, поведінку, оцінки, переваги і результати виконання навчальних завдань, запити тощо).

Цю інформацію, подану відповідно до існуючих стандартів, можуть автоматизовано обробляти інтелектуальні ПА. Особливо важливо досягти інтероперабельності у поданні знань різних ПрО, що забезпечує повторне використання навчальних матеріалів.

SCORM (Shareable Content Object Reference Model) - промисловий стандарт для обміну навчальними матеріалами на базі концептуальної моделі стандарту IEEE 1484.12.1

Мета створення SCORM – забезпечення багаторазового використання навчальних матеріалів, інтероперабельності навчальних курсів, супроводу й адаптації курсів, поєднання інформації окремих навчальних матеріалів у навчальні курси або дисципліни відповідно до індивідуальних запитів користувачів.

Основою моделі SCORM є модульна побудова навчального матеріалу. Модулі навчального матеріалу в SCORM називаються SCO (Shareable Content Objects). SCO – автономна одиниця навчального матеріалу, що має метадані і змістовну частину. Сукупність модулів певної ПрО називають бібліотекою знань (Web-репозиторієм). Модулі SCO можуть у різних комбінаціях поєднуватися один з одним у складі навчального матеріалу. Для цього створюють LMS – систему керування керування контентом.

У SCORM для подання вмісту модулів використовується мова XML, визначаються зв’язки з програмним середовищем і API, надаються специфікації створення метаданих, що базуються на стандарті IEEE 1484.12.1.

Слід зазначити, що SCORM базується на ще одному стандарті програмного доступу до інформаційного навчального ресурсу IEEE 1484.11.2 Standard for Learning Technology – ECMA Script Application Programming Interface for Content to Runtime Services Communication, однак різні країни реалізують неоднакові види програмного інтерфейсу: приміром, у Канаді – за допомогою з'єднання peer-to-peer; в Австралії – за допомогою SOAP. Тому для доступу до навчальних ІР має використовуватися посередник – ПА навчального ресурсу, що реалізує різні моделі доступу до ресурсів.

Зараз існує багато інструментів дистанційного навчання та *e*-learning, які різняться за функціями та цілями [150]. Призначення ПА в цій сфері – забезпечити слухачам доступ до потрібних курсів та зручні умови навчання. Так, студенти різних спеціальностей вчаться за різними програмами і в багатьох випадках мають різну теоретичну і практичну підготовку. Їх персональні ПА аналізують це і пропонують їм не тільки універсальну програму курсу, але і додаткові факти і довідники від інших курсів, яких вони не вивчали.

Через те, що процес навчання безпосередньо пов'язаний з обміном знаннями між викладачем та студентом, ПА мають використовувати формалізоване інтероперабельне подання знань. Сьогодні найбільш поширені системи, що використовують для цього онтології. Приміром, у [298] пропонується мультиагентний онтологічний підхід до створення розподілених систем дистанційного навчання (СДН), що базується на принципах проекту Semantic Web і агентних технологіях. Використання стандартів технологій дистанційного навчання (ДН) забезпечує доступ до територіально розосереджених навчальних ІР , створених різними постачальниками освітніх послуг та поданих в Інтернеті, і функціонування СДН в єдиному інформаційно-освітньому середовищі.

Характерною рисою архітектури мультиагентної онтологічної СДН (МОСДН) є реалізація розподіленості і персоналізації за допомогою мультиагентного онтологічного підходу. Розподіленість забезпечується за рахунок використання ПА, територіально розосереджених на різних комп'ютерах: персональні агенти створюються для кожного, кого навчають, на порталі ДН, агент-координатор здійснює керування СДН на сервері ДН, агент навчальних ІР здійснює доступ до навчальних матеріалів з комп'ютерів різних постачальників освітніх послуг. Персоналізація навчання досягається за рахунок онтологічних моделей, що забезпечують метазнання для здійснення індивідуального підбору навчальних матеріалів з використанням МОСДН. Ці метазнання подані мовою OWL.

У [210] докладно описується використання ПА в проекті реорганізації електронної бібліотеки, у якому реалізовані ПА навчального ІР і онтологічна модель цього ІР для здійснення доступу до ресурсу в електронній бібліотеці.

Мобільні ПА можуть використовуватися для пошуку у репозиторіях навчальних об'єктів [276]. ПА виконує роль посередника, забезпечуючи в МАС процес пошуку акторів й

навчальних ІР. Посередник зберігає профіль користувача (статистику його звернень до різних репозиторіїв). Практична реалізація забезпечує взаємодію з репозиторієм навчальних об’єктів, що відповідає стандартам IMS Digital Repositories Interoperability, IMS Metadata, IMS Content Packaging та частково – ADL SCORM. Мовою запитів є Xquery. Система побудована на платформі Microsoft .NET мовою C#.

У [266] пропонується реалізація МАС для підтримки сервера дистанційного навчання з психолінгвістики. Метою роботи є побудова мережі інтелектуальних ПА, які підтримують працездатність сервера дистанційного навчання (htt://www.csa.ru/DistanceLearning) – збирають інформацію в Інтернеті, структурують її і подають користувачам у зручній формі.

Система складається з п’яти ПА. Координатор виконує роль дошки оголошень для інших ПА. Саме через цього агента можна запросити необхідну послугу або з’ясувати адресу конкретного ПА. DB-агент – ПА БД забезпечує доступ до інформації в ній, а також відповідає за всю роботу, проведену з цією інформацією. *Search*-агент призначений для пошуку інформації в мережі. Побачивши нове слово, Search-агент звертається до звичайних пошукових систем, відбирає інформацію і, переглянувши її, щоб упевнитися у тім, що ця інформація стосується теми, записує Інтернет-адресу в БД. *Parser*-агент забезпечує більш глибоке пророблення інформації в БД. Побачивши в БД непроороблене посилання, Parser-агент, як і Search-агент, звертається до пошукових систем і за тим же алгоритмом відбирає чергові посилання. Робота Search-агента і Parser-агента з БД ведеться через DB-агента. *Client*-агент забезпечує зв'язок користувачів з інформацією, збереженою в БД. Користувач вводить термін, а ПА повертає йому всю інформацію, збережену в БД. Введення нових термінів до БД передбачається тільки експертом.

Запити ПА подаються мовою KQML, а знання, якими вони обмінюються, – мовою KIF. Для реалізації МАС використовуються мови програмування JAVA і Python. Робота над БД організується за протоколом JDBC.

Дистанційна освіта – новий засіб організації освітнього процесу, що базується на принципі самостійного навчання студента.

Розроблена у Курському державному університеті система дистанційної освіти (СДО) базується на основі мультиагентної технології. СДО містить засоби навчання, контролю практичних завдань, а також віддаленного спілкування з викладачем.

Для створення СДО ефективним виявилося використання мультиагентного підходу, у рамках якого система будується як сукупність ПА (агенти користувача, викладача, лекцій, окремих об'єктів знання). Кожний з агентів має семантичний опис свого поля діяльності, переслідує власні цілі, обмінюється інформацією з іншими ПА для досягнення компромісів тощо.

В основу реалізації підходу покладено семантичні мережі знань. Метавузлами цієї мережі є такі об’єкти: лекція, лабораторна робота, тестове завдання, додаткова література. Дані вузли можуть бути зв'язані відношеннями типу "*потрібно для розуміння*", "*спирається на*", "*близька тема*", "*рекомендується для подальшого вивчення*". Кожен вузол може мати атрибути типу "*складність маршруту*", "*час вивчення*" тощо. За допомогою формальних операцій над графом ПА викладача може сформувати оптимальний маршрут вивчення матеріалу для заданої мети, рівня підготовки тощо. Для узгодження цих параметрів ПА викладача має можливість обмінюватися інформацією з ПА учня, знаходячи компроміс.

У процесі роботи користувача МАС має можливість одержувати більшу кількість даних про його переваги як явно (анкетування, обробка користувальницьких запитів), так і неявно (наприклад, аналізуючи статистику відвідування різних розділів). На базі цієї інформації можна будувати евристичні класифікації користувачів і припущення про "наступні кроки" користувача, відповідним чином перебудовувати засоби навігації, формувати освітні сценарії (наприклад, залежно від рівня підготовки користувача або часу).

У системі можна виділити три типи ПА: агент користувача, викладача, квантів знань (лекцій, контрольних питань, лабораторних робіт тощо). Будь-який запит користувача через гіперпосилання активізує серію переговорів між агентами. На першому етапі переговори здійснюються між ПА квантів знань і агентом користувача, виявляючи в такий спосіб на цьому кроці елементи знань системи, що необхідні користувачеві. Аналіз відбувається на основі семантичної мережі знань, з вузлів якої утворяться агенти квантів знань, що володіють зв'язками і відношеннями між собою, і інформації (моделі переваг користувача і статистики відвідування вузлів семантичної мережі), про яку знає персональний агент користувача.

Після того, як зі списку квантів знань, що утримуються в системі, відібрано необхідну для користувача кількість, починається другий етап переговорів між ПА викладача й користувача. На цьому

кроці агент викладача з відібраних на першому етапі лекцій, контрольних питань, лабораторних робіт формує оптимальний маршрут навчання для конкретного користувача, на основі його рівня підготовки, заданої мети тощо. Агент викладача також формує і рекомендації для користувача, скориставшись якими, можна змінити маршрут навчання і спрямувати його іншим шляхом.

Застосування концепції інтелектуальних ПА для розробки системи дистанційного утворення дозволяє спростити і якісно поліпшити процес одержання людиною знань і інформації, дає можливість персональному агентові користувача виконувати автономно задачі, поставлені перед ним, здобувати і систематизувати знання, що дозволить вивести подібні системи на якісно інший рівень, зробивши агентів незамінними помічниками в процесі навчання.

Наприклад, в Уфімському державному авіаційному технічному університеті створено систему автоматизованого дистанційного навчання, призначену для проведення занять зі студентами різних форм навчання (очного, вечірнього, заочного). Система є розподіленою і будується за схемою “Центр навчання – пункти навчання”.

Взаємодія між компонентами системи здійснюється з використанням мережі Інтернет. «Інтелектуальне ядро» системи реалізовано як МАС і включає набір ПА, зокрема інтелектуальних, що взаємодіють між собою за заздалегідь описаними правилами в складній людино-машинній системі. ПА системи здатні посилати запити і одержувати відповіді, виконуючи інформаційний обмін з іншими віртуальними агентами системи. Інтелектуальні ПА здійснюють обробку навчально-методичної та іншої інформації, що зберігається в БД системи, її адаптацію щодо переваг користувачів і можливості транспортної підсистеми і доставку її користувачам. Використання агентів забезпечує особливо високу гнучкість, ефективність, а також надійність обробки інформаційних потоків в інформаційному освітньому середовищі. Для зміни сценаріїв роботи системи достатньо змінити або доповнити базу правил функціонування інтелектуальних програмних агентів. Для розширення функціональних можливостей системи в неї може бути додані чергові ПА, для чого необхідно визначити як правила їх функціонування, так і порядок і правила взаємодії з уже існуючими в системі ПА.

Сценарії поведінки інтелектуальних ПА проектуються з урахуванням можливості адаптації щодо переваги конкретних

тьюторів, типу навчання з використанням дистанційних технологій.

Взаємодіючи з користувачами системи, інтелектуальні ПА збирають, накопичують і потім обробляють дані про їх переваги. Це дозволяє ПА змінювати сценарії своєї поведінки, що зрештою приводить до самонастроювання системи під вимоги конкретних користувачів – слухачів і тьюторів. Підбір даних може здійснюватися у вигляді обробки запитів, аналізу траєкторії переміщення в навчальному просторі і анкетування.

Для організації зберігання і обробки навчально-методичного матеріалу використовується об'єктно-орієнтований підхід з розміщенням структурованої навчально-методичної інформації в базі даних системи дистанційного навчання. Навчально-методичний матеріал є незалежною МАС дистанційного навчання. Стандартизація форматів подання навчальних матеріалів на базі закладених в систему шаблонів, дозволяє поліпшити сприйняття інформації, що надається користувачу, а також швидко змінювати інтерфейсні рішення шляхом зміни шаблонів.

4.6. Програмні агенти електронного урядування

Перспективним напрямком застосування агентних технологій є *e*-government (електронне урядування).

Посередниками у взаємодії між урядом і громадянами держави можуть стати ПА, які спрощують та пришвидшують цей процес, персоніфікуючи його для окремих громадян. Наявність зрозумілої та достовірної моделі ПА, що підтримують відповідні сервіси, підвищує ймовірність звертання до них.

На сьогодні існує велика кількість застосунків МАС у цій сфері. Приміром, у роботі [172] розглядається персональний асистент для підтримки користувачів – державних службовців, що звертаються до МАС *e*-government. Цей асистент має інтелектуальну архітектуру. У ньому використовуються онтології для обробки знань та підтримки діалогу природною мовою. ПА у системі подаються як Web-сервіси. Основна ціль такого підходу – запропонувати ІС, здатну отримувати завдання через інтуїтивно зрозумілий інтерфейс, накопичувати знання та дозволяти недосвідченим користувачам взаємодіяти з системою просто і комфортабельно.

При звертанні користувача до МАС *e*-government створюється персональний ПА цього користувача. Цей агент інтерпретує запит користувача т звертається до агента сервісу електронного урядування, який відсилає його до сутності або множини сутностей, які

впроваджують відповідні компоненти сервісів *e*-government за повноваженнями держави. Наприклад, працівники уряду та співпрацюючі з ними провайдери Web-сервісів спільно розповсюджують ліцензії водіїв, ліцензії для рибальства та мисливства на певній території [170].

Вивчення намірів населення щодо використання сервісів електронного урядування у складних і катастрофічних ситуаціях й факторів, що їх визначають, наводиться у [118].

Загальні критерії довіри – компетентність, доброзичливість і чесність. Громадяни не захочуть залежати від агента сервісу *e*-government, якщо вони переконані у тому, що цей ПА не здатний забезпечити сервіс, який він пропонує, не зацікавлений у суспільному добробуті або не готовий гарантувати мінімальний рівень виконання сервісу. Різні громадяни мають неоднаковий рівень довіри до ПА сервісів *e*-government, і це визначає різний рівень намірів використання цих сервісів.

4.7. Програмні агенти електронної медицини

Електронна медицина (*e*-медицина) покриває весь спектр медичних процесів та сервісів. Розвиток ІТ дозволив збільшити її популярність у різноманітних сферах: *e*-діагностика, *e*-фармацевтика, *e*-охорона здоров'я, телемедицина тощо. Мультиагентний підхід можна використовувати і в системах *e*-медицини.

E-медицина має багато сфер застосування – від діагностики до телемоніторингу [88] та телеконсультацій. Застосунки *e*-медицини можна поділити на чотири категорії [213]:

- охорона здоров’я та медицина;
- персоналізована інформація про стан здоров’я;
- телеконсультації спеціалістів;
- продовження медичної освіти.

E-медицина має значний вплив на традиційну систему охорони здоров’я на рівні країни або регіону. Для її успішного використання та повної реалізації потенціалу, який вона має, потрібно застосовувати сучасні ІТ [221].

Мультиагентний підхід широко застосовується для створення великих складних систем. Автономність та соціальні властивості ПА є базисом для цього. Саме тому доцільно базувати складні системи е-медицини на агентних технологіях.

Медицина взагалі виконує чотири базові функції:

- профілактика хвороб;

- діагностика хвороб;
- лікування хвороб;
- консультування здоров'я.

Відповідно вимоги до систем е-медицини поділяються на функціональні (дистанційне медичне обслуговування і клінічна медична практика, ведення медичних БД, обмін медичною інформацією тощо) та нефункціональні (надійність та приватність).

Е-медицина (телемедицина, медична освіта та адміністрування) безпосередньо пов'язана з поданням в електронній формі медичних процесів та сервісів. Зараз значна частина інформації про пацієнтів, що проходять лікування в лікарні, подається в електронній формі, що дозволяє використовувати для її обробки комп'ютерною технікою. Варто враховувати, що ця інформація надходить з різних джерел через різні інтервали часу, подається у різних видах і може мати великий обсяг (для графічних зображень, мультимедіа, аудіофайлів тощо).

Е-фармацевтика фокусується на стандартизації та розвитку традиційної фармацевтики, особливо – біомедицини.

Телемедицина – це використання телекомунікаційних технологій для підтримки охорони здоров'я. Телекомунікаційні ІС мають забезпечувати медичне обслуговування населення, що поширене на значних територіях (приміром, у сільській місцевості) та розподіляти сучасний медичний досвід та технологічну базу через телекомунікації та ІТ. Як приклад підтримки нових сервісів телемедицини можна навести проект M2DM.

Ціль проекту M2DM – забезпечення цілодобового нагляду за хворими на діабет [74]. Цей проект створює нове інформаційне середовище спілкування між пацієнтами та лікарями. Архітектура системи містить два типи ПА: агентів комунікаційного сервера, які взаємодіють з різноманітними терміналами користувачів і агентів сервера застосунків, які обробляють і аналізують отримані дані.

Електронна охорона здоров'я пов'язана з підтримкою діяльності клінік та має використовувати МАС для встановлення взаємодії між спеціалістами-медиками та пацієнтами, обміну медичною інформацією. Існують дві моделі електронної охорони здоров'я – асинхронна та реального часу.

Електронна діагностика спрямована на аналіз та визначення стану здоров'я пацієнтів. Дані про пацієнтів (CPR – Computer-based patient record) накопичуються у БД в електронній формі. Такі дані

можуть бути подані спеціалістам у різному вигляді – тексти, таблиці, графіки, аудіозаписи, зображення, мультимедіа [115].

Призначення е-медицини – поширити найновіші методи лікування для все більшої кількості людей. У БД лікарень та інших закладів охорони здоров'я зберігаються величезні масиви даних. Можна використовувати цю інформацію для здобуття знань, обміну досвідом. Але це потребує її стандартизації та формалізації. У побудові ефективних лікувальних процесів та заходів можна виділити такі одиниці:

- медичний технологічний процес;
- ознака захворювання;
- імператив (вказівка);
- оператор, тобто активна частина лікувального заходу, яка призводить до зміни стану пацієнта.

Лікувальний захід складається з наборів обмежень, імперативів та операторів. Виконання операторів може бути як послідовним (тобто порядок виконання операторів впливає на їх результат), так і паралельним (оператори можуть виконуватися у довільній послідовності). Слід відзначити специфіку медичного технологічного процесу – його незворотність, тобто більшість операторів не мають зворотного.

Різноманіття взаємопов'язаних та розподілених ресурсів, властиве медичним даним і задачам, робить агентну парадигму ефективною для впровадження у медичних системах, так само як і у сфері менеджменту пацієнтів [88, 116].

Мультиагентний підхід потребує визначення ролей ПА в системі е-медицини, ідентифікації їх можливостей та сервісів, які вони надають. Можна виділити два кроки у розробці МАС – розробку окремих ПА та організацію співтовариства агентів.

Визначимо ролі ПА в системі е-медицини. Інтерфейсний ПА забезпечує користувачам доступ до МАС. Одним з інтерфейсних агентів є персональний помічник, який ініціює запит користувача до системи та подає його результати в зручній формі. Агент лікаря є персональним помічником певного лікаря і підтримує розклад його роботи та візитів пацієнтів. Агент-брокер знає всі можливості МАС і забезпечує іншим ПА доступ до потрібних їм сервісів.

Адміністративний ПА призначений для адміністрування діяльності МАС та забезпечує кооперацію між різними відділами та їх ПА. Контролюючий ПА здійснює нагляд за роботою інших агентів та вирішує конфлікти між ними.

Агент відділення контролює діяльність системи е-медицини на рівні певного відділення і використовує знання, що відповідають його специфіці.

Крім того, МАС е-медицини може містити агентів моніторингу, а також спеціалізованих ПА – терапевтів, діагностів, хірургів, консультантів тощо

Крім того, у такій системі зазвичай є ПА для роботи з БД, для підтримки прийняття рішень, для навчання.

МАС *е*-медицини мають тенденцію використовувати гібридну архітектуру., що поєднує переваги реактивної та деліберативної. Різні МАС (приміром, МАС різних клінік) можуть взаємодіяти одна з одною, забезпечуючи як передачу даних про пацієнтів, так і знання персоналу.

ПА може обробляти дані про пацієнтів неперервно, відстежуючи зміни в його стані. Інтелектуальний ПА може допомогти лікарю в таких питаннях:

- відстежувати критичні стани хворих та інформувати про це лікаря;
- узагальнювати знання експерта про методи діагностування для поліпшення якості діагностики;
- прогнозувати наступні стани пацієнта на основі даних його обстеження;
- навчатися з досвіду роботи лікаря

У цьому напрямку агенти можуть застосовувати методи ШІ (експертні системи, ймовірнісні мережі, індуктивне узагальнення тощо) [245].

Розглянемо приклад системи е-медицини, яка надає підтримку для віддаленої взаємодії між лікарями, які використовують спільні ІР. Ця система базується на агентно-орієнтованому засобі JAMES (Java Agent Model for Enhanced Services), на основі Java [62]. Найважливішою рисою такої МАС є здатність до взаємодії. Для взаємодії між ПА використана мова KQML, а для метаописів мультимедійних документів – RDF . Комунікації здійснюються через Інтернет, за протоколами TCP/IP, SMTP та FTP, а також KQML. Для керування мультимедійними даними використовується бібліотека HYPERFRAME, яка подає відомості про документи через RDF.

MANTHA (Multi Agent Network for Telematics and Hypermedia Applications) [44] – агентно-орієнтована архітектура керування гіпермедійними документами. Спеціалізується у телепатології. Спочатку використовувалась для простого обміну електронними листами про випадки патологій для експертизи. Зараз її можливості

значно розширені. ПА, що входять до складу системи, забезпечують керування БД, розподіл ресурсів, інтероперабельність пошти та інших сервісів мереж.

Переваги використання телемедицини [28]:

- гнучка організаційна структура;
- більша доступність охорони здоров’я;
- підвищення якості охорони здоров’я завдяки можливості швидко застосовувати найкращі засоби незважаючи на географічне розташування;

зниження вартості медичних послуг завдяки економії на зменшенні масштабу.

Висновки

Сучасний етап розвитку ІТ визначається двома основними тенденціями: обробкою знань та застосуванням відкритих систем. Саме на перетині цих напрямків виникли агентні технології. Активний розвиток методів і технологій розподіленого штучного інтелекту, досягнення в сфері апаратних і програмних засобів підтримки концепції розподіленості і відкритості спричинили розвиток мультиагентних систем, в яких ПА спільно вирішують складні завдання.

Здатність програмних агентів автономно планувати та координувати свої дії, вести переговори з іншими розподіленими застосунками в складному гетерогенному інформаційному середовищі, гнучко і інтелектуально приймати рішенння в динамічно змінюваних і непередбачуваних ситуаціях приводить до того, що агентно-орієнтоване програмування стає ключовою технологією обробки інформації.

Ще одна важлива властивість програмних агентів – персоніфікована та інтелектуальна взаємодія з користувачем. Навчаючись у процесі взаємодії з користувачем та іншими агентами, агент здатний більш ефективно задовольняти потреби користувача та визначити свою стратегію співпраці з іншими агентами. Розвиток агентних технологій дозволяє зробити новий крок у взаємодії користувача з ІС, ціль якого – використання знань кінцевих користувачів для більш якісного задоволення їх інформаційних потреб та автономного виконання рутинних операцій.

У цій роботі, звичайно, неможливо торкнутися усіх питань, пов’язаних з проектуванням, аналізом та використанням програмних агентів та мультиагентних систем. Агентні технології – область, що швидко розвивається та змінюється. Тому автори намагалися розкрити відносно сталі принципові теоретичні аспекти агентної парадигмита проілюструвати їх прикладами архітектур та ПЗ, які можуть допомогти у розробці застосунків з різноманітних областей.

Список літератури

1. A Model-Theoretic Semantics for DAML+OIL, W3C Note 2001 http://www.w3.org/TR/daml+oil-model.
2. Accessibility Features of SVG, W3C Note 2000 http://www.w3.org/TR/SVG-access/.
3. *Adorani G., Poggi A.* An object-oriented language for distributed artificial intelligence. International Jornal of Man-Machine Studies, 38:435-453, 1993.
4. Agent Construction Tools. – http://www.agentbuilder.com/ AgentTools/index.html.
5. Agent software. – http://www.agentlink.org.
6. Agent sowtware examples.– http://www.agents/tools/index.html.
7. AIFB Web Page Ontology. – http://ontobroker.semanticweb. org/ontos/aifb.html.
8. *Agha G., Chapman D.* PENGI: An implementation of a theory of activity. Proc.of the 6th National Conference on Artificial Intelligence. 1987. – P.268-272.
9. *Alexander J.H., Freilin M.J., Shulman S.J., Rehfuss S., Messick S.L.* Knowledge level engineering: ontological analysis // Proc.National Conf. on Artificial Intelligence, 1986. – P.963-968.
10. Amalthaea. – http://reference.allrefer.com/encyclopedia /A/Amalthae.html
11. A Model-Theoretic Semantics for DAML+OIL, W3C Note 2001 http://www.w3.org/TR/daml+oil-model.
12. Applying machine learning to semiconductor manufacturing / K.B.Irani, J.Cheng, U.M.Fayad, Z.Qian // IEEE, 1993. – V.2. – P.41-47.
13. AuctionBot.– http://auction.eecs.umich.edu.
14. Autonomy's Kenjin. – http://www.kenjin.com.
15. *Balabanovic M.*. An adaptive web page recommendation service. In Proceedings of the First International Conference on Autonomous Agents, ACM Press, New York, 1997. – P. 378-385.
16. BargainFinder. – http://bf.cstar.ac.com/bf.
17. *Bates J.* The role of emotiom in believable agents. Communications of the ACM. N37. – 1994. – P.122-125.
18. *Berners-Lee T.* Business Model for the Semantic Web – Design Issues, 2001 http://www.w3.org/DesignIssues/Business.
19. *Bechhofer S., Horrocks I., Goble C., Stevens R.* OilEd: A Reason-able Ontology Editor for the Semantic Web // Joint German/Austrian

conf. on Artificial Intelligence (KI'01). Lecture Notes in Artificial Intelligence LNAI 2174, Springer-Verlag, Berlin, pages.396-408, 2001.
20. *Berners-Lee T., Hendler J., Lassila O.* A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. – http://www.w3.org/2001/sw.
21. *Bichanan B.G., Mitchell T.M.* Model-directed learning of production rules // Pattern-directed Inference Systems / Ed.D.A. Waterman, F.Hayes-Roth. – New York: Academic Press, 1978. – P.24-58.
22. *Bigus J., Bigus J.* Constructing Intelligent Agents Using Java: Second Edition. – New York: Addison-Wesly, 2002. – 520 p.
23. *Bratman M.E.* What is intention? In P. R. Cohen, J. L. Morgan, and M. E. Pollack, eds.. Intentions in Communication,. The MIT Press: Cambridge. MA, 1990. P. 15-32
24. *Bratman M. E., Pollack M. E.* Toward an architecture for resource-bounded agents. Technical Report CSLI-87-104, Center for the Studv of Language and Information, SRI and Stanford University, August 1987.
25. *Bray T.* RDF and Metadata, 1998. – http://www.xml.com/xml/pub/98/06/rdf.html
26. *Bray T., Paoli J.* Spenberg-McQeen C.M. Extensible Markup Language (XML) 1.0 Cpecification. – W3C. –1998. – http://www.w3.org/TR/REC-xml.
27. *Brooks R.A.* Intelligence without reason. In Proc. of the Twelfth International Joint Conf. on Artificial Intelligence (IJCAI-91), Sydney, Australia, 1991. – P.569-595.
28. *Bui T., Sankaran S.* Software Agents for Telemedicine. – http://www.vacets.org/vtic97/txbui2.htm.
29. *Busuioc M., Winter C.* Negotiation and Intelligent Agents, Project NOMADS-001, BT Labs, Martlesham Heath, U.K., 1995.
30. *Bunneister B., Sundermeyer K.* Cooperative problem solving guided by intentions and perception. In E. Wernerand Y. Demazeau, eds.. Decentralized AI 3 – Proc. of the Third European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds, P. 77-92. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1992.
31. *Buntine W.* Decision Tree Induction Systems. A Bayesian Analisys // Machine Intelligence and Pattern Recognition. – 1987. – Vol.6. – P.109-128.
32. *Castelfranchi C.* Social power. Decentralized AI – Proceedings of the

First European Workshop on Modelling Authonomous Agents in Multi-Agent Worlds, 1990. – P.49-62.
33. *Chakrabarti S., Berg M., Dom B.* Focused crawling: a new approach to topic-specific Web resource discovery. – 1999, Elsevier. – P.545-562.
34. *Chakrabarti S., Dom B., Agrawal R. , Raghavan P.* Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies // The VLDB Jornal, 1998. – P.163-178.
35. *Chapman D.* Planning for conjunctive goals. // Artificial Intelligence, N.32, P.333-378, 1987.
36. *Cheong F.-C.* Internet Agents: Spiders, Wanderers, Brokers and Bots, New Riders Publishing, Indianapolis, 1996.ISBN: 1-56205-463-5.
37. CLIPS. – http://www.ghg.net/clips/CLIPS.html.
38. *Codd E.F.* Extending the Datebase Relational Model to Capture More Meaning // ACM Transactions on Datebase Systems. – 1979. – Vol.4, N.4. – P.397-434.
39. *Cohen P.R., Levesque H.J.* Intention is choice with commirment. Artificial Intelligence. N 42. 1990. – P.213-261.
40. *Cohen P.R., Levesque H.* Persistence, intention and commitment. In M. Georgeffand A.L. Lansky, editors. Proceedings of the 1986 Workshop on Reasoning About Actions and Plans, pages 297-340. Morgan Kaufmann Publishers, 1987.
41. Copernic Software. – http://www.copernic.com/.
42. *Dastani M., Jacobs N., Jonker C.* Modeling User Preferences in Agent-Mediated Electronic Commerce. – Springer Verlag, 2000.
43. Deep Web – Bright Planet's white paper. – http://www.completeplanet. com/tutorials/deepweb.
44. *Della Mea V., Beltrami C.A.* Telepathology applications of the Internet Multimedia Electronic Mail // Medical Informatics 23, 1998. – P.237-244.
45. *Dennett D.C.* The Intensional Stance. The MIT Press: Cambridge, MA, 1987. – 282 p.
46. Describing and retrieving photos using RDF. – http://www.w3.org/TR/photo-rdf/.
47. *Dignum F.* Agents, markets, institutions and protocols. – Springer Verlag, 2000.
48. Distributed XML: the role played by XML in the next-generation Web, – http://www.xml.com/pub/2000/09/06/distributed.html.
49. DOE – The Differential Ontology Editor. – http://opales.ina.fr/public/.

50. *Domingue J. Tadzebao* WebOnto: Discussing, Browsing, and Editing Ontologies on the Web // Proc. of the Eleventh Workshop on Knowledge Acquisition, Modeling and Management, KAW'98, Banff, Canada, 1998.
51. DTD standardizations, e.g. HR-XML. – http://www.hr-xml.org/.
52. Dublin Core. – http://dublincore.org.
53. Dublin Core Metadata for Resoerce Discovery. – http://www.faqs.org/rfcs/rfc2413.html.
54. EchoSearch – http://jdj.sys-con.com/read/35754.htm.
55. Editors Assessment. – http://www.semwebcentral.org/ assessment/report.
56. *Ermolayev V.* RACING: Agent-Mediated Web Service Composition for Rational Information Retrieval. – http://www.zsu.zp.ua/racing.
57. *Ermolayev V., Keberle N., Plaksin S., Vladimirov V.* Ontology-Driven Query Transformation in Agent-based Intelligent Information Retrieval. Herald NTU KhPI, Sp. Issue "System Analysis, Control, and IT", N 1, 2004. – P. 53-66.
58. Evolution Ontology. – http://kaon.semanticweb.org/exemples/ Evolution.rdfs.
59. Firefly.- http://www.firefly.com.
60. FuzzyCLIPS. – http://ai.iit.nrc.ca/fuzzy/fuzzy.html.
61. IEEE Standart Upper Ontology. – http://suo.ieee.org/.
62. Internet Agents for Telemedicine Services / V.D.Mea, V. Roberto, A.Conti, L.Gaspero, C.A.Beltrami. – Medical Informatics & Internet in Medicine (formerly Medical Informatics) 1999. – P.179-186.
63. *Chavez Anthony and Maes Pattie* Kasbah: An Agent Marketplace for Buying and Selling Goods // Proceedings ofPAAM'96. - Practical Applications Company, 1996.
64. *Genesereth M. R., Ketchpel S.P.* Software agents // Communications of the ACM. – 37(7), 1994. – P.48-53.
65. Hypertext Transfer protocol (HTTP) // Gettis J., Mogul J., Frystyk H., Berners-Lee T., MasinterL., Leach P. 1999. – http://www.3w.org/Protocols/rfc2616/rfc2616.html.
66. *Guarino N.* Formal Ontology and Information Systems // National Research Council, Proceedings of FOIS'98, Trento, Italy, ─ June 1998. ─ Amsterdam, IOS Press. – P. 3-15.
67. KA2 Science Ontology. – http://ontobroker.Semanticweb. org/ontos/ka2.html.
68. *Ermolayev V., Plaksin S.* Cooperation Layers in Agent-Enable Business Process Management // Проблемы программирования, №

1-2, 2002. – C.354-367.
69. *Eriksson J., Finne N., Janson S.* SICS MarketSpace – an agent-based market infrastructure // SICS Technical Report. – Swedish Institute of Computer Science, February 1998.
70. *Farquhar A., Fikes R., Rice J.* The Ontolingua server: A tool for collaborative ontology construction // International Journal of Human-Computer Studies, 46(6), , 1997. – P.707–728.
71. *Ferguson I.A.* TouringMachines: an Architecture for Dynamic, Rational, Mobile Agents. Technical Report N.273, University of Cambridge Computer Laboratory, 1992.
72. *Fernandez M, Gomez-Perez A., Pazos J. A* Building a Chemical Ontology Using Methondology and the Ontology Design Environment // IEEE Intelligent Systems, Jan./Feb. pages 37-46, 1999.
73. *Fikee R. E., Hart P. E., Nilsson N.* STRIPS: A New Approach to the Application of Theorem Proving // Artificial Intelligence, 2. – P.189-208.
74. Final Report. M2DM: multi-access services for the management the diabetes melltus. EC project. – http://www.aim.unipv.it/projects/M2DM .
75. *Finin O., Fritzson R.* KQML – a language and protocol for knowledge and information exchange. In Proceedings of the 11th Intl. Distributed Artificial Intelligence Workshop, page» 127-136, Seattle, WA, USA, 1994. – P.127-136
76. FIPA – Federation of Intelligent Physical Agents. Home Page. – http://www.cselt.stet.it/fipa/fipa_rationale.htm.
77. FireFly Network. – www.firefly.net/.
78. *Forbus K. D., Feltovich P. J.* The Coming Revolution in Educational Technology. Smart Machines in Education, eds. Forbus, K. D. and Feltovich, P. J., , AAAI Press/MIT Press, 2001. – P.3-5.
79. *Forgy C.L.* OPS5 User`s Manual, Department of Computer Science. – Pittsburgh: Carnegie-Mellon University, 1981. – 226 p.
80. Foundation for Intelligent Physical Agents. FIPA 98 Specification. Part 1. Agent Management. – http://www.fipa.org.
81. *Franklin A., Graesser A.* Is it an agent or just a programm: a taxonomy form autonomos agents. / In Proc. of the III Int. Workshop on Agents Theories, Arch. and Languages./N.-Y., Springer-Verlad, 1996.
82. *Georgeff M.P., IngrandF.F.* Decision-making in an embedded reasoning system. In Proc. of the Eleventh International Joint Conf. on Artificial Intelligence (IJCAI-89), 1989. – P. 972-978.

83. IDEFS Method Report. – www.idef.com/IDEF5.html.
84. Jango Shopbot. – www.jango.com.
85. *Jennings N.R.* Specification and implementation of a belief desire joint-intention architecture for collaborative problem solving. // J. of Intelligent and Cooperative Information Systems, v.2, N.3, 1993. – P.289-318.
86. *Gillmor D.* Small portals prove that size matters, Tech column in San Jose Mercury News, December 1998, http://www.sjmercury.com/columnists/gillmor/docs/dg120698.htm.
87. *Gladun A., Rogushina J.* Multiagent Ontology-Based Intelligent System Of E-Commerce // Proceedings of Int.Conf. TPSD`2004, Kiev. – P.55-58.
88. *Hayes-Roth B., Larsson, J.E.* A domain-specific software architecture for a class of intelligent patient monitoring systems. Journal of Experimental and Theoretical Artificial Intelligence 8, 1996. – P.149-171.
89. *Hayes-Roth F., McDermott J.* An inference matching technique for inducing abstractions // Comm. ACM. – 1978. – Vol. 21, N.5. – P.401-410.
90. *Hersovici M., Jacovi M., Maarek Y.S. , Pelleg D. , Shtalheim M., Ur S.* The Shark-Search algorithm — an application: tailored Web site mapping // 7th World-Wide Web Conference, April, 1998, Brisbane, Australia. – ttp://www7.scu.edu.au/programme/fullpapers/1849/com1849.htm.
91. Gaia Methodology. – http://www.ecs.soton.ac.uk/~nrj/download-files/jaamas2000.pdf.
92. *Gallier J.R.* A Theoretical Framework for Computer Models of Cooperative Dialue, Acknowledging Multi-Agent Conflict.PhD thesis, Open University, UK, 1988. – 46 p.
93. *Geissler C., Konolige K.* A resolution method for quantified modal logics of knowledge and belief. In J.Y.Halpern, ed.. Proc. of the 1986 Conf. on Theoretical Aspects of Reasoning About Knowledge, P.309-324. Morgan Kaufmann Publishers: CA, 1986.
94. *Georgeff M.P., Lansky A.L.* Reactive reasoning and planning. In Proc. of the Sixth National Conf. on Artificial Intelligence (AAAI-87), Seattle, WA, 1987. – P. 677-682
95. *Georgeff M.P., Rao A.S.* Formal model and Decision Procedures for Multi-Agent Systems. Technical note 61, Australian Artificial Intelligence Institute, 1995.
96. *Gilbert D.J.* IBM Intelligent Agent Strategy. / IBM Corporation, 1995.

97. *Ginsberg M.* Knowledge interchange format: The KIF of death // AI Magazine, 1991. – P.24-29.
98. *Gladun A., Rogushina J.* Multiagent ontology-based intelligent system of e-commerce // Вісник Київського національного університету ім.Т.Шевченко. Серія "Фізико-математичні науки", 2004. - P.118-124.
99. *Glover E., Lawrence S., Birmingham W., Giles C.L.* Architecture of a metasearch engine that supports user information needs. In Eighth International Conference on Information and Knowledge Management, CIKM 99, Kansas City, Missouri, November 1999. – P.210-216.
100. *Gruber T.* Ontolingua: a Mechanism to Support Portable Ontologies. Knowledge Systems Laboratory of Stanford University. – www.gruber92ontolingua.pdf.
101. *Hayes-Roth F., McDermott J.* An inference matching technique for inducing abstractions // Comm. ACM. – 1978. – V. 21, N.5. – P.401-410.
102. Heterogeneos agent systems / V.S.Subrahmanian The MIT Press. Cambridge, Massachusetts, London, England, 2000. – 580 p.
103. *Hintikka J.* Knowledge and Belief. – Cornell University Press: Ithaca. NY, 1962.
104. IDEFS Method Report. – www.idef.com/IDEF5.html.
105. *Jennings N.R.* Cooperation in Industrial Multi-agent Systems. World Scientific Publishing. 1994.
106. *Jennings N.R., Campos J.R.* Towards a Social Level Characterisation of Socially Responsible Agents. – Software Engineering, IEE Processing, V.144, 1997. – P.11-25.
107. IEEE 1484.12.1 Standard for Learning Object Metadata, PISCATAWAY, NJ, 2002. - p. 44.
108. *Kaelbling L.P., Rosenschein S.J.* Action and planning in embedded agents. In P.Maes, ed., Designing Autonomous Agents, The MIT Press: Cambridge, MA, 1990. – P. 35-48.
109. Kasbah. – https://kasbah.media.mit.edu.
110. *Kay A.* Computer Software // Scientific American, V. 251, №.2, 1984. – P. 53-59.
111. *Kifer M., Lausen G, Wu J.* Logical Foundations of Object-Oriented and Frame-Based Languages // Jornal of ACM. – 1995.
112. KnowledgeWright. – http://www.softkey.info/reviews/review.php
113. *Konolige K. A* Deduction Model of Belief. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1986. -226 p.

114. *Kripke A.* Semantical analysis of modal logic. Zeilschriftfiir Mathematische Logik and Grundlagen der Mathematik, № 9, 1963. – P.67-96.
115. *Kun L. G.* Telehealth and the global health network in the 21st century. From homecare to public health informatics. Computer Methods and Programms in Biomedicine, Issue 64, 2001. – P. 155–167.
116. *Lanzola G., Falasconi S., Stefanelli M.* Cooperative software agents for patient management. Lecture Notes in Artificial Intelligence 934, 1995. – P.173-184.
117. *Lawrence S., Giles C.* Searching the World Wide Web. Science, 280(5360), 1998. – P. 98-100.
118. *Lee L., Braynov S., Rao H.* Effects of a public emergency on citizens' usage intention toward e-government: A study in the context of war in Iraq. – www.cse.buffalo.edu/doc/1243.htm.
119. *Lee Raymond S.Tand, Liu James N.K.* iJADE eMiner - A Web-Based Mining Agent Based on Intelligent Java Agent Development Environment (iJADE) on Internet Shopping. – Springer Verlag, 2000.
120. Lee R. S. T. and Liu J. N.K.: FAgent - An Innovative E-Shopping Authentication Scheme using Invariant Intelligent Face Recognition Agent. – Proc. of International Conference on Electronic Commerce (ICEC2000) , Seoul, Korea. – P.47-53.
121. Leppinen M., Rautiainen A. Agents in Commerce.– http://smartpush.cs.hut.fi/Software Agents_in_Commerce/.
122. *Letichevsky A., Gilbert D.* A model for interaction of agents and environments. WADT'99. – www-lsr.imag.fr/WADT99/Abstracts/07.html.
123. *Levesque H. J.* A logic of implicit and explicit belief. In Proc. of the Fourth National Conf. on Artificial Intelligence (AAAI-84), P. 198-202, Austin, TX, 1984.
124. *Lieberman H.* An Agent That Assists Web Browsing. – http://lieber.media.mit.edu.
125. *Lienhart R., Pfeiffer S., Effelsberg W.* Video Abstracting. – Comm.ACM, Vol.40, N.12, 1997. – P.54-62.
126. *Liu J., Ye Y.* E-Commerce Agents, LNAI 2033, Springer-Verlag Berlin Heidelberg, 2001.– P. 1-6.
127. *Lomusciot A., Wooldridge M., Jennings N.* Automated Negotiation in Agent-Mediated Electronic Commerce. – Springer Verlag, 2000.
128. *Maes P.* The Agent Network Architecture. – SIGART Bulletin, 2(4),

1991. – P.112-120.
129. *Makinson D.* How to give it up: a survey of some formal aspects of theory change // Synthese. – 1985. – Vol.62, N.3. – P.325-383.
130. *Magedanz T.* On the Integration of IN and TMN – Modeling IN-based Service Control Capabilities as part of TMN-based Service Management, Chapman & Hall, London (GB), Santa Barbara, California (USA), 1995. – 387 p., (http://www.fokus.gmd.de/research/cc/oks/uni/.
131. *Magedanz T.* Intelligent Agents: State of the Art and Potential Application Areas in Future Telecommunications. – Springer Verlag, Berlin , 1995. – 365 p.
132. MAS-Common KADS. – http://portal.acm.org/citation.cfm?id=749430.
133. MaSE Methodology. – http://www.pa.icar.cnr.it/~cossentino/al3tf1/docs/mase4agentlink.pdf.
134. *McArthur D., Lewis M., and Bishay M.* The Roles of Artificial Intelligence in Education: Current Progress and Future Prospects. 1993. - http://www.rand.org/hot/mcarthur/ Papers/role.html.
135. *McCarthy J.* Ascribing mental qualities to machines. Techn.report, Stanford University Al Lab., Stanford, 1978.
136. Metadata Architecture, W3C Design Issues. – http://www.w3.org/DesignIssues/Metadata.
137. *Michalski R.S., Garbonell J.G., Mitchell T.M.* Machine Learning // An Artificial Intellegence Approach. – Tioga: Palo Alto, CA, 1983. – 344 p.
138. Microsoft Proofing Tools – www.bmsoftware.com/microsoftproofingtools.htm.
139. *Miller E.* An Introduction to the Resource Description Framework, 1998 http://www.dlib.org/dlib/may98/miller/05miller.html.
140. MIME. – http://www.isi.edu/in-notes/iana/assignments/media-types/media-types.
141. *Minsky M.* Progress Report on Artificial Intelligence // Seymour Papert ec 11, 1971. – http:// web.media.mit.edu/~minsky/papers/PR1971.html.
142. *Moore R.C.* A formal theory of knowledge and action. Readings and planning. San Mateo. Ca. 1990. P.480-519.
143. MPEG: achievements and current work.. – http://www.cselt.it/mpeg/terms_of_reference.htm.
144. MPEG-1, ISO/IEC, 1996. – http://mpeg.telecomitalialab.com/standards/mpeg-1/mpeg-1.htm

145. MPEG-21 Multimedia Framework, Introduction, ISO/IEC. – http://mpeg.telecomitalialab.com/standards/mpeg-21/mpeg-21.htm.
146. MPEG-21 Overview, 2002. – http://mpeg.telecomitalialab.com/standards/mpeg-21/mpeg-21.htm.
147. MPEG-7 Overview. 2002. – http://mpeg.telecomitalialab.com/standards/mpeg-7/mpeg-7.htm.
148. *Muller J. P.* The Design of Intelligent Agents: a layered approach. – Springer, 1996. – 219 p.
149. *Muller J. P., Pischel M.* Modelling interacting agents in dynamic environments. In Proc. of the Eleventh European Conf. on Artificial Intelligence (ECAI-94), Amsterdam, The Netherlands, 1994. – P. 709-713.
150. *Murray T. , Blessing S. , Ainsworth S.* Authoring tools for advanced technology learning environments: towards cost-effective adaptive, interactive, and intelligent educational software. – http://helios.hampshire.edu/~tjmCCS/atoolsbook/chaptersV2/ChapterList.html.
151. *Musen M.* Domain Ontologies in Software Engineering: Use of Protege with the EON Architecture // Methods of Inform. in Medicine, 1998. – P.540-550.
152. *Myerson R.* Game Theory: Analysis of Conflict. Harward University Press, Cambrige, Massachusetts. 1991.
153. *Ndumu D.T., Nwana H.S.* Research and Development Challenges for Agent-Based Systems. – Software Engineering, IEE Processing, V.144, 1997. – P.2-10.
154. *Negroponte N.* Agents: from direct manipulation to delegation. / In Software Agents. 1997.
155. Netscape on OPS. – http:developer.netscape.com/ops/ops.html.
156. NeurOK Semantic Suite. – http://soft.neurok.ru/products/semantic.shtml
157. *Newell A.* The knoledge level. – Artificial Intellegence, N 18, 1982. – P. 87-127.
158. *Newell A., Simon H.A.* Computer science as empirical enquiry. // Comm. of the ACM, v.19, 1976. – P.113-126.
159. *Noy N., Musen M.* The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping // Stanford Medical Informatics, Stanford Univ., 2003.
160. *Nute D.* Defeaseble Reasoning and Decision Support Systems // - Decision Support Systems. - 1988. - Vol.4, N.1, - P.97-110.
161. *Nwana H., Lee J., Jennings N.* Coordination in software agent

systems. / British Telecommunication Technology Journal, 14(4), 1996.– P.79-88.
162. *Nwana H., Wooldridge M.* Software Agent Technologies. – http://www.labs.bt.com/projects/agents/publish/papers/.
163. OKBC: A Programmatic Foundation for Knowledge Base Interoperability. V. Chaudhri, A. Farquhar, R. Fikes P. Karp J. Rice // Fifteenth National Conf. on Artificial Intelligence. AAAIPres/The MIT Press, Madison, 1998. – P.600-607.
164. *Oliveira E., Rocha A.* Agents Advanced Features for Negotiation in Electronic Commerce and Virtual Organisation Formation Process. – Springer Verlag, 2000.
165. OntoEdit: Collaborative ontology development for the Semantic Web. Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, D. Wenke // In Proc. of the Inter. Semantic Web Conference (ISWC 2002), Sardinia, Italia, June 2002.
166. OntoSeek: Content-Based Access to the Web – http://dunwell.computer.org.
167. Open Directory Project. – http://dmoz.org/.
168. *Osipov G.S.* Semantic Types of Natural Language Statements. A Method of Representation // 10th IEEE International Symposium on Intelligent Control, 1995. – P.285-291.
169. OWL Web Ontology Language 1.0 Reference, W3C Working Draft 29 July 2002. – http://www.w3.org/TR/2002/WD-owl-ref-20020729/.
170. *Palmer I.* State of the World: E-Government Implementation," Faulkner Information Services, 2003 – www.faulkner.com/products/faulknerlibrary/00018297.htm.
171. *Pan J.Y., Tenenbaum J.M.* An intelligent agent framework for enterprise integration. IEEE Transactions on Systems, Man and Cybernetics, 21(6), 1991.
172. *Paraiso E.C., Barthès J.-P.* A Voice-Enabled Assistant in a Multi-Agent System for e-government Services. – www.vea.project/docs/35467.htm.
173. *Paurobally S., Gunningam J.* Formal Models for Negotiatians Using Dynamic Logic. – Springer Verlag, 2000.
174. *Pawlak Z.* Knowledge and uncertainty. A rough set approach/ Proc. of workshop "Incompleteness and Uncertainty in Information Systems". – Oct. 8-9, 1993. – P. 1-11.
175. PersonaLogic. – http://www.personalogic.com.
176. *Petrie C.J.* Agent-based ingeneering, the WEB and intelligent. / IEEE

expert, 11(6): 1996. – P.24-29.
177. PICS Rating Vocabularies in XML/RDF, W3C NOTE 2000. – http://www.w3.org/TR/rdf-pics.
178. *Plotkin G.* Note on inductive generalization // Machine Intelligence. – 1971. – Vol.5. – P.18-47.
179. *Plotkin G.* A further note on inductive generalization // Machine Intelligence. – 1971. – Vol.6. – P.6-54.
180. Precision Graphics Markup Language (PGML) W3C Note 1998. – http://www.w3.org/TR/1998/NOTE-PGML-19980410.html.
181. *Preece A., Hui K., Gray P.* KRAFT: Supporting Virtual Organizations by Constraint Fusion. – www.csd.abdn.ac.uk/research/kraft.html.
182. Processing of Incomplete Data in Example-Learning Systems / N.I.Galagan, J.V.Rogushina, E.I.Nechvalenko, E.N.Borovaya // Proc. of EWAIC, Sept. 7-9. – Moscow, 1993, – P. 301-305.
183. Programming Expert Systems in OPS5: An Introduction to Rule-Based Programming / L.Brownston, R.Farrel, E.Kant, N.Martin. – Ney-York: Addison-Wesley, 1986. – 187 p.
184. *Quinlan J.R.* Discovery rules from large collections of examples: a case study // Expert Systems in the Microelectronic Age. – Edinburg, 1979. – P.87-102.
185. *Quinlan J.R.* Learning Efficient Classification Procedures and Their Application to Chess End Games // Machine Learning: An Artificial Intellegence Approach / R.S.Michalski, J.G.Carbonell, T.M. Mitchell. - Tioga, Palo Alto, 1983. - P.463-482.
186. *Raggett D., Hors A.L., Jacobs I.* HTML 4.01 Specification. W3C Recommendations. 1999. – http://www.w3.org/TR/html401.
187. *Rao A.S., Georgeff M.P.* A model-theoretic approach to the verification of situated reasoning systems. In Proc. of the Thirteenth international Joint Conf. on Artificial Intelligence (IJCAI-93), Chambery, France, 1993. – P. 318-324.
188. *Rao A.S., Georgeff M.P.* Modeling rational agents within a BDI-architecture. In R. Pikes and E. Sandewall, eds.. Proc. of Knowledge Representation and Reasoning (KR&R-91), Morgan Kaufmann Publishers: San Mateo, CA, April 1991. – P. 473-484.
189. RDF Editor. – http://www.xmlspy.com.
190. RDF Query Language (RQL). – http://139.91.183.30: 9090/RDF/VRP/index.html/RQL/index.html.
191. RDF Schema – RDFS. – http://www.w3.org/TR/PR-rdf-schema.
192. RDF Site Summary RSS. – http://groups.yahoo.com/group/rss-

dev/files/schema.rdf.
193. RDF syntax, W3C Recommendation. – http://www.w3.org/TR/PR-rdf-syntax.
194. RDFT. – http://www.cs.vu.nl/~borys/RDFT/0.27/RDFT.rdfs.
195. Reichgelt H. Logics for reasoning about knowledge and belief. Knowledge Engineering Review, 4(2): 119-139, 1989.
196. Requirements for a Web Ontology Language, W3C Working Draft http://www.w3.org/TR/webont-req/.
197. *Riechem D.* Intelligent Agents // Communication of the ACM, 1994, Vol. 37, No. 7, P.20-21.
198. RETSINA Calendaring Agent. – http://ilrt.org/discovery/ 2001/06/schemas/icalfull/hybrid.rdf.
199. *Rhodes B. J., Starner T.* Remembrance Agent: A continuously running automated information retrieval system. In Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi Agent Technology, 1996. – P.487-495.
200. *Rouge D.* et al. The Semantic Grid: a Future e-ScienceInfrastructure. – http://www.semantic grid.org/documents/ semgrid-jornal/semgrid-jornal.pdf.
201. *Russel S.J.* Artifical Intelligence: A modern approach. Prentice Hall, Upper Saddle River, New Jersey, 1995. – 685 p.
202. *Rzevski G.A.* Multi-Agents Logistics and E-commerce // MAGENTA White Paper, 1999. – 14 p.
203. *Sacerdoti E.* Planning in a hierarchy of abstraction spaces // Artificial Intelligence, N.5, 1974. – P.115-135.
204. Scalable Agent-based Information Retrieval Engine (SAIRE) – www.hq.nasa.gov/hpcc/reports/annrpt97/accomps/iita/WW227.htm.
205. Scalable Vector Graphics, W3C Candidate Recommendation, 2000. – http://www.w3.org/Graphics/SVG.
206. Semantic Translation, http://www.ecimf.org/contrib/onto/ST/ index.html.
207. Sesame. – http://sesame.aidministrator.nl/.
208. *Shoham Y.* Agent-oriented programming. Artificial Intelligence. N 60. 1993. – P.51-92.
209. *Skowron A.* Management of Uncertainty in AI: A Rough Set Approach.// Proc. of workshop "Incompleteness and Uncertainty in Information Systems", Oct. 8-9, 1993. – P. 36-61.
210. Software agents for learning resources of digital library / Milashenko D.A., Makovetskiy S.D. , Boblovskiy R.V. , Keleberda I.N. , Lesna N.S. // Proceedings of 2nd International Conference ISTA'2003.

Lecture Notes in Informatics. - Bonn: Kolen Druck+Verlag GmbH., 2003. – P.77-84.
211. Standard Generalized Markup Language SGML-standart. – http://www.iso.ch/cate/d16387.html.
212. *Subrahmanian V.S.* Heterogeneos agent systems. – The MIT Press, Cambridge, Massachusetts, London, England, 2000. – 580 p.
213. *Suleiman A. B.* The untapped potential of telehealth. International Journal of Medical Informatics, Issue 61, 2001. – P.103–112.
214. SWAD-Europe: Mapping Semantic Web Data with RDBMSes. – http://www.w3.org/2001/sw/Europe/reports/scalable_rdbms_mapping_report/.
215. *Sycara K.* Multiagent systems – http://www.aaai.org/Pathfinder/html/ multiagent_systems.htm.
216. Synchronized Multimedia Integration Language (SMIL 2.0), W3C Recommendation, 2001. – http://www.w3.org/TR/smil20/.
217. Agents. – Caput Technical Computer Science. – http://www.cs. rug.nl/ben/agents/trading-agents.http.
218. Tete-a-Tete. – http://ecommerce.media.mit.edu/tete-atete.
219. The DRM WebWatcher – http://www.disabilityresources. org/DRMwww.html.
220. The Upper Cyc Ontology. – http://www.cyc.com/cyc-2-1/index.html.
221. *Tian J., Schillo H.* A Multi-agent Approach to the Design of an E-medicine System: MATES 2003, LNAI 2831, © Springer-Verlag Berlin Heidelberg, 2003. – P. 85–94.
222. Towards Automating Recognition of Differing Problem-Solving Demands / J. Stoute, G. Caplain, S. Marcus, J. McDermott // AAAI WorkShop on Knowledge Acquisition for Knowledge-Based Systems. – 1987. – P.894-900.
223. UM-PRS: An Implementation of the Procedural Reasoning System for Multirobot Applications / J.Lee, M.J. Huber, E.H.Durfee, and P.G.Kenny. In Conference on Intelligent Robotics in Field, Factory, Service, and Space (CIRFFSS'94), 842-849, Houston, Texas, 1994.
224. *Uschold M., Gruninger M.* Ontologies: Principles, methods and applications // Knowledge Engineering Review. ─ 1996..─ Vol. 11, No. 2. ─ P.93–155.
225. *Utgoff P.E.* An Incremental Induction of Decision Trees. – Alliherst: Massachusets University, 1989. – 174 p.
226. *Vaschenco N.* Operation on Concepts and their Realization when Knowledge Formation // Моделирование творческих процессов на основе баз знаний. – София, 1992. – P.19-33.
227. Vector Markup Language (VML), W3C Note 1998. –

http://www.w3.org/TR/NOTE-VML.
228. *Vere S.A.* Multilevel Counterfactuals for Generalizations of Relational Concepts and Productions // Artificial Intellegence. – 1980. – Vol.14, N.2. – P.139-164.
229. *Vere S., Bickmore T.* A basic agent. // Computational Intelligence, N.6, 1990. – P.41-60 .
230. *Vetter M., Pitsch S.* Towards a Flexible Trading Process over the Internet. – Springer Verlag, 2000.
231. *Viamonte M., Ramos C.* A Model for an Electronic Market Place – Springer Verlag, 2000.
232. VRML 1.0 Specification, W3C Note, ISO/IEC 14772-1:1997. – http://www.web3d.org/technicalinfo/specifications/vrml1.0.htm.
233. W3C The Semantic Web Home Page. – http://www.w3.org/ 2001/sw/
234. W3C Web Ontology. – http://www.w3.org/2001/sw/WebOnt/.
235. WebCompass Page. – http://www.symantec.com/techsupp/ webcompass/kbase_webcompass.html.
236. WebSeeker 5.0. – www.bluesquirrel.com/products/webseeker/.
237. Web Services Description Language (WSDL) Version 1.2, W3C Working Draft 2003. – http://www.w3.org/TR/2003/WD-wsdl12-20030124/.
238. *Wielinga B.J., Breuker J.A.* Models of expertise // Proc. Of 7th European Conf. Of Artificial Inelligence, 1986. – P.306-318.
239. *Wooldridge M.* An Introduction to MultiAgent Systems. – http://www.csc.liv.ac.uk.
240. *Wooldridge M.* The Logical Modelling of computational multi-agent systems. Technical report MMU-DOC-94-01, Manchester, 1992.
241. *Wooldridge M., Jennings N.* Agent Theories, Architectures, and Languages: A Survey. In: Intelligent Agents. ECAI-94 Workshop on Agent Theories, Architecture and Languages. Amsterdam, The Netherlands, August 8-9, 1994, (Eds. M.J.Wooldridge and N.R.Jennings). Proceedings. Springer Verlag: 3-39, 1994.
242. *Wooldridge M., Jennings N.* The cooperative problem solving process: A formal model. Technical report, Department of Computing, Manchester Metropolitan University , Chester St., Manchester M1 5GD, UK, 1994. 15.
243. *Wooldrige M., Jennings N.* Intellegent Agents: Theory and Practice / Knowledge Engineering Rewiew, Vol.10, No.2, 1995.P.115-152.
244. WordNet / EuroWordNet. – http://www.cogsci.princeton.edu/~wn.
245. World Health Organization. Technical report, – http://www.who.int, http://www.med.hokudai.ac.jp/ seniorw/Oth-

lec/htele/htele2/006.htm.
246. *Ygee F.* Software Agent for Electronic Power Trade. – www.enersearch.se/ygge.
247. *Андон Ф.И., Яшунин А.Е., Резниченко В.А.* Логические модели интеллектуальных информационных систем. – Киев: Наукова думка. – 1999. – 401 с.
248. *Андон Ф.И., Яшунин А.Е., Резниченко В.А.* Логическое направление интеллектуализации информационных систем // Праці Першої міжнар. наук.-практ. конф. з програмування. – 1998. – С. 368–380.
249. *Атанасова Т.Б.* Агентна технология: концепции, модели, приложения. – Варна, 2000.
250. *Бабенко Л.П., Лаврищева К.М.* Основи програмної інженерії.- К.: Знання, 2001. – 269 с.
251. *Бандурка О.М., Тягло О.В.* Курс логіки. Підручник. – К.: Літера ЛТД. – 2002. – 160 с.
252. *Боровикова О.И., Загорулько Ю.А., Сидорова Е.А.* Автоматизация сбора онтологической информации в Интернет-портале знаний // 5 Международная конференция "Интеллектуальный анализ информации ИАИ-2005". – К.: Просвіта, 2005. – С.82-91.
253. Мультиагентная система моделирования производства и продажи автомобилей / С.В.Батищев, К.В.Ивкушкин, И.А.Минаков, Д.А.Ржевский,П.О.Скобелев – http://www. madi.ru/logistics/ccl/resourses/sts/08/08.htm.
254. *Брусенцов Н.П.* Дедукция, диалектика и аристотелево содержательное следование, Искусственный интеллект, 2000, №2, – С.285-291.
255. *Брукшир Д.Г.* Введение в компьютерные науки.- М.: Вильямс, 2001. – 688 с.
256. *Валькман Ю.Р., Золотаревский И.А., Квачев В.Г., Яковенко Л.П.* Распределенный искусственный интеллект и многоагентные системы проектирования сложных изделий // Проблемы программирования, №2, 1999. – С.75-85.
257. *Варшавский В.И., Поспелов Д.А.* Оркестр играет без дирижера. М: Наука 1984.
258. *Васильев С.Н.* От классических задач регулирования к интеллектуальному управлению // Известия Академии наук. Теория и системы управления. – 2001, № 2. – С.5-21.
259. *Вассерман Л. И., Дюк В. А., Иовлев Б. В., Червинская К. Р.*

Психологическая диагностика и новые информационные технологии. — СПб.: СЛП, 1997.
260. *Вельбицкий И.В.* Графический стиль и стратегия профессиональной технологии программирования // Тез. докл. II Всесоюзн. конф. "Технология программирования". – Киев: Ин-т кибернетики им.В.М.Глушкова АН УССР, 1986. – С.14-32.
261. *Верников Г.* Стандарт онтологического исследования IDEF5. – www.vernikov.ru/material36.html.
262. *Виттих В.А., Скобелев П.О.* Разработка мультиагентной системы для моделирования процессов принятия решений в компаниях с сетевой организацией. – http://eup.kulichki.net/Documents/2002-08-05/F176.htm
263. *Виттих В.А., Скобелев П.О.* Мультиагентные системы для моделирования процессов самоорганизации и кооперации. – http://eup.kulichki.net/Documents/2002-08-05/F172.htm.
264. *Гаврилова Т. А., Червинская К. Р.* Извлечение и структурирование знаний для экспертных систем. — М.: Радио и связь, 1992.
265. *Гаврилова Т.А., Хорошевский В.Ф.* Базы знаний интеллектуальных систем. – СПб.: Питер, 2001.
266. *Гаврилова Т.А., Яшин А.М., Фертман В.П.* Взаимодействие интеллектуальных агентов для поддержки сервера дистанционного обучения. – http://www.inftech.webservis.ru/it/conference/isanditc/2000/section3/rus/arrus4.html.
267. *Галаган Н.И., Барецкая Л.П., Рогушина Ю.В.* Интегрированный интеллектуальный комплекс ЭКОН // Тез. докл. I междунар. конф. "Технология программирования 90-х". – Киев, 1992. – С.44-46.
268. *Герасимова И.А.* Логический анализ рассуждений на основании личностных знаний // Синтаксические и семантические исследования неэкстенциональных логик. – М.: Наука, 1989 – С.50-66.
269. *Гладун А.Я., Несен М.В., Рогушина Ю.В.* Агентно-орієнтований підхід при вирішенні задач е-комерції та пошуку інформації // міжнародна науково-практична конференція "Розробка систем програмного забезпечення: виклики часу та роль в інформаційному суспільстві", Київ, 2005. – С.75-78.
270. *Гладун А.Я., Рогушина Ю.В.* Multiagent ontology-based intelligent system of e-commerce // Вісник Київського національного університету ім.Т.Шевченка. Серія "Фізико-математичні науки", 2004. - С.118-124.

271. *Гладун В.П.* Планирование решений. – Киев: Наук. думка, 1987. – 168 с.
272. *Гладун В.П.* Процессы формирования новых знаний. – София, 1994. – 189 с.
273. *Гладун В.П.* Эвристический поиск в слож-ных средах. – Киев: Наук. думка, 1977. – 166 с.
274. *Гладун В.П., Величко В.Ю., Киселева Н.Н., Москалькова Н.М.* Вывод гипотез о составе и свойствах объектов на основе аналогии // Искусственный интеллект, 2000, №1. – С.44-52.
275. *Глибовець М.М., Олецький О.В.* Штучний інтелект. – К.: Вид.дім "КМ Академія". – 2002. – 366 с.
276. *Глибовець М.М., Постніков О.С.* Використання мобільного агента при пошуку в розподілених репозиторіях навчальних об'єктів / Вісник Київського національного університету ім.Т.Шевченка. Серія "Фізико-математичні науки", 2004. - С.128-130.
277. *Городецкий В.И., Грушинский М.С., Хабалов А.В.* Многоагентные системы (обзор) // Новости искусственного интеллекта. – N 2, 1998. – С.64-116.
278. *Городецкий В.И., Лебедев А.Н.* Планирование и составление расписаний автомати-ческого удовлетворения ограничений на временную структуру процесса // Проблемы информатизации. – N3- 4, 1994. С. 49-55.
279. *Гришанова І.Ю., Рогушина Ю.В.* Аналіз засобів подання, контексту та метаописів мультимедійних інформаційних ресурсів для підвищення релевантності їх пошуку // Проблеми програмування, № 3, 2005. – С.31-44.
280. *Грищенко В.Н., Лаврищева Е.М.* Компонентно-ориентированное программирование. Состояние, направление и перспективы развития // Проблемы программирования. – 2002, № 1-2. – С. 80-90.
281. *Гупал А.М., Цветков А.М.* Инкрементные алгоритмы индуктивного вывода // Разработка и использование информационных технологий в системах управления – Киев: Ин-т кибернетики им.В.М.Глушкова АН Украины, 1993. – С. 105-112.
282. *Дерецкий В.А.* Об одном подходе к обработке естественно-языковых данных на основе анализа семантических сетей // Матеріали І Міжнародної науково-практичної конференції з програмування УкрПрог'98, Київ, 1998. – С.405-411.

283. *Джексон П.* Введение в экспертные системы. – С.: Изд.дом "Вильямс", 2001. – 624 с.
284. Диалоговые системы и представление знаний/Кокорева Л.В., Перевозчикова О.Л., Ющенко К.Л.-К.:Наукова думка,1992. – 448 с.
285. *Дубинский А.Г.* Автоматизація інформа-ційного пошуку в глобальній мережі. // Харк. держ. політехн. ун-т. Вісник. Вип. 73. Системний аналіз, упр. і інформ. технології – Харків: ХДПУ, 1999. – С. 54-59.
286. *Дубинский А. Г.* Характеристики эффекти-вности информационного поиска в сети Интернет. - http://dubinsky.nm.ru/pub/01s01/01s01.htm.
287. Дюк В., Самойленко А. Data mining: учебный курс. – СПб: Питер, 2001. – 368 с.
288. *Ермолаев В. А., Плаксин С. Л.* Координация делегирования работ в коалициях агентов, выполняющих задания. – http://eva.zsu.zaporizhzhe.ua/eva_personal/PS/cwp-ZSU-25-07.pdf.
289. *Заде Л.А.* Понятие лингвистической переменной и его применение к принятию приближенных решений. – М.: Мир, 1976. – 165 с.
290. *Золотова Г.А., Онипенко Н.К., Сидорова М.Ю.* Коммуникативная грамматика русского языка.– М.: Азбуковник, 2004. – 324 с.
291. *Ивашко В.Г., Кузнецов С.О.* Оценки правдоподобия в продукционных экспертных системах // В сб. Экспертные системы: состояние и перспективы. – М.: Наука, 1989. – С.92-103.
292. *Ивашко В.Г., Финн В.К.* Экспертные системы и некоторые проблемы их интеллектуализации. – Семиотика и информатика.– М.: 1986. – В.27. – 208 с.
293. Интеллектуальные агенты в Интернете. – http://webagents.report.ru/_5FolderID_25_.html.
294. Интеллектуальный семантический поиск с привлечением средств метапоиска // Осипов Г. С., Завьялова О.С., Смирнов И. В., Тихомиров И. А. – 5 Международная конференция "Интеллектуальный анализ информации ИАИ-2005". – К.: Просвіта, 2005. – С.214-223.
295. Искусственный интеллект. В 3-х кн. Кн.2. Модели и методы: Справочник / Под ред. Д.А. Поспелова. – М.: Радио и связь, 1990. – 303 с.
296. *Каменнова М.* Управление электронными документами:

технологии и решения – http://www.osp.ru/os/1995/04/38.htm.
297. *Картышева Е.* Интеллектуальные поисковые системы Excalibur. – http://www.osp.ru/nets/1997/06/98.html.
298. *Келеберда И.Н., Лесная Н.С., Репка В.Б.* Использование мультиагентного онтологического подхода к созданию распределенных систем дистанционного обучения. – http://ifets.ieee.org/russian/depository/v7_i2/html/3.html.
299. *Козлов Д.Д., Смелянский Р.Л.* Использование интеллектуальных агентов для поиска информации в Интернет // Искусственный интеллект, №2, 2002. – С.378-382.
300. *Кокорева Л.В., Перевозчикова О.Л., Ющенко К.Л.* Диалоговые системы и представление знаний. – К.: Наукова думка,1992.– 448 с.
301. *Крипке С.А.* Семантическое рассмотрение модальной логики // Семантика модальных и интенсиональных логик. – М.: Прогресс, 1981. – С. 27–40.
302. *Кузьменко Г.Є., Литвинов В.А.* Прагматичний підхід до оцінки рівня інтелекту інтелектуалізованих систем // Математичні машини і системи. – 2003, № 1. – С.3-9.
303. *Лефевр В.А.* Конфликтующие структуры. – М.: Институт психологии РАН, 2001. – 265 с.
304. Логический подход к искусственному интеллекту: от классической логики к логическому программированию / Тейз А., Грибомон П., Луи Ж. и др. – М.: Мир, 1990. – 432 с.
305. Лорьер Ж.-Л. Системы искусственного интеллекта. – М.: Мир, 1991. – 568 с.
306. *Лоуренс С.* Контекст при поиске в Web // Открытые системы. – №12, 2000. – http://www.osp.ru/os/2000/12/.
307. *Любіч О.О., Плескач В.Л., Рогушина Ю.В.* Інформаційна підтримка управління державними фінансами засобами штучного інтелекту // Вісник КНТЕУ, №3, 2004. - С.90-95.
308. *Любич А.А., Плескач В.Л., Рогушина Ю.В.* О выборе критериев оценки интеллектуальности информационной системы // УсиМ, № 1, 2005. – С.3-7.
309. *Матыс Е.Г., Некрасова И.Ю.* Повышение конкурентоспособности предприятия на основе контроля качества продукции. – http://conf.susu.ru/doc/menedg/mats.shtml.
310. *Милль Д.С.* Система логики силлогической и индуктивной. – М., 1900. – 176 с.
311. *Минский М.* Фреймы для представления знаний. – М.: Энергия,

1979. – 151 с.
312. Моделирование слабоформализуемых предметных областей с неполными данными с применением индуктивного вывода / Н.И.Галаган, Ю.В. Рогушина, Э.Н.Боровая, Е.И.Нечваленко. // Тез. докл. II Международного научно-технического семинара "Теоретические и прикладные проблемы моделирования предметных областей в системах баз данных и знаний" / Под ред. Игнатенко Б.В. (Туапсе, 20-26 сентября 1993 г.). – Киев, 1993. – С.119-124.
313. Модель мультиагентной системы для е-бизнеса и технология ее программной реализации / Гриценко В.И., Гладун А.Я., Журавлев Ю.Д., Несен М.В.// Проблемы программирования, № 2-3, 2004. – С.510-519.
314. *Моросов М.* Модели и методы решения задач. – http://khpi-iip.mipk.kharkiv.edu/library/ai/conspai/02.html.
315. *Москалькова Н.М.* Обчислювальна схема виведення за аналогією для структурно-атрибутивних моделей // Проблеми програмування, 1999, №1. – С.100-105.
316. Мультиагентная система для решения задач логистики / К.В. Ивкушкин, И.А. Минаков, Г.А. Ржевский, П.О. Скобелев. – http://www.madi.ru/logistics/ccl/resourses/sts/07/07.htm.
317. Мультиагентная система корпоративных новостей / С.Батищев, О.Лахин, И.Минаков, Г.Ржевский, П.Скобелев. – http://www.kg.ru/Publish/ artic41.htm.
318. *Нетесин И.Е., Рогушина Ю.В.* Методика повышения актуальности распределенной базы знаний // УСиМ, N 4-5, 1996, С.108-112.
319. *Овдій О.М., Проскудіна Г.Ю.* Обзор инструментов инженерии онтологий. – http://www.elbib.ru/rus/journal/2004/part4/op.html.
320. *Овдій О.М., Проскудіна Г.Ю.* Онтології у контексті інтеграції інформації: представлення, методи та інструменти побудови // Проблемы программирования. – №4, 2004. – С.353-366.
321. *Осуга С. Обработка знаний.* – М., Мир, 1989. – 324 с.
322. *Паринов С.И., Яковлева Т.И.* Экономика 21 века на базе Интернет-технологий. – http://rvles.ieie.nsc.ru/parinov/economy21.htm.
323. *Плескач В.* Мультиагентні системи в електронній комерції // ВЦ КНТЕУ, Вісник КНТЕУ, №6, 2003. – С.35-41.
324. *Плескач В.Л.* Онтологии в контексте представления знаний об электронном бизнесе // Экономика: проблемы теории и

практики. Т.3, 2004. – С.697-702.
325. *Плескач В.Л., Рогушина Ю.В.* Декларация пространств имен ресурсов электронного бизнеса для идентификации домена // Проблемы программирования, №2-3, 2004. – С.366-369.
326. *Плескач В.Л., Рогушина Ю.В.* Применение метода анализа иерархий для сравнения и выбора систем электронного бизнеса // Вісник Київського національного університету ім.Т.Шевченко. Серія "Фізико-математичні науки", 2004. - С.156-163.
327. *Плескач В.Л., Рогушина Ю.В.* Розробка засобів інтероперабельності електронного бізнесу на основі XML // Вісник КНТЕУ, № 1, 2004. - С. 89-95.
328. *Плескач В.Л.* Технология Web-сервисов для электронного бизнеса // НАУ, збірник наукових праць "Проблеми підвищення ефективності інфраструктури ", №8, 2002. – С.188-193.
329. *Плескач В.Л.* Технології електронного бізнесу. - К.: Київ.нац.торг.-екон.ун-т, 2004. – 222 с.
330. *Плескач В.Л., Рогушина Ю.В., Кустова Н.П.* Інформаційні технології та системи. - підручник МОНУ. К.: Київ.нац.торг.-екон.ун-т, 2004. – 520 с.
331. *Плескач В.Л. , Рогушина Ю.В.* Применение java-технологий и xml для реализации приложений в области электронного бизнеса // Комп'ютерні засоби, мережі та системи, №3, 2004. – С.140-148.
332. Поисковые агенты. – http://www.vn.iatp.org.ua/web/a/30/myweb/iseekag.htm.
333. *Поляков А. О.* Технология интеллектуальных систем: Учеб. пособие. – СПб.: СПбГТУ, 1995. – 242 с.
334. *Пономаренко Л.А., Плескач В.Л., Рогушина Ю.В.* Критерии оценки уровня интеллектуальности информационных систем // Вестник НТУ "ХПИ", збірник наукових праць, випуск №18, 2003. – С.111-122.
335. *Пойа А.* Математика и правдоподобные рассуждения. – М.: ИЛ, 1964. – 189 с.
336. *Протасова Э.Н., Рогушина Ю.В.* Расширение модальной логики для формализации интенсиональных отношений. – УСиМ, 1999, №4. – С. 32-36.
337. *Раудис Ш.Ю.* Алгоритмы построения правила классификации (обзор) // Статистические проблемы управления. – Вильнюс: Ин-т физики и математики Лит.ССР, 1973. – Вып. 2. – С.12-49.
338. *Ржевский Д.* Мультиагентные системы в логистике и е-

коммерции. – http://www.ec-logistics.ru/articles/sts/09/09.htm.
339. *Рогушина Ю.В.* Индуктивный вывод – средство формирования базы знаний прикладной экспертной системы // Материалы научн.-техн. семинара "Программное обеспечение новых информационных технологий" – Тверь,1991. – С.63-65.
340. *Рогушина Ю.В.* Инструментарий для разработки прикладных экспертных систем в слабоформализованных областях ИндЭкс 2.0 // Тез.докл.семинара "Новые информационные технологии и инструментально-технологические средства принятия решений" (Кацивели, 21-26 декабря 1992 г.). – Киев: Ин-т кибернетики им.В.М.Глушкова, 1992. – С. 101-102.
341. *Рогушина Ю.В.* Использование онтологического описания предметной области для повышения релевантности информационного поиска // Проблемы программирования, №4, 2003. – С.54-64.
342. *Рогушина Ю.В.* Разработка средств интеллектуализации поиска информации в Интернет // Проблеми програмування, №1-2, 2002. – С.378-385.
343. *Рогушина Ю.В., Боровая Э.Н.* Использование индуктивных методов для извлечения знаний из неполных данных // Программная инженерия. Сб.научн.тр. – Киев: Ин-т программных систем, 1994. – С.31-40.
344. *Рогушина Ю.В., Боровая Э.Н., Нечваленко Е.И.* Методика извлечения знаний из примеров в системах, обрабатывающих неполные данные // Представление знаний в информационных технологиях. – Киев: Ин-т кибернетики им.В.М.Глушкова АН Украины, 1993. – С. 60-65.
345. *Рогушина Ю.В., Гладун А.Я.* Засоби інтелектуалізації пошукових механізмів в Інтернет // 5 Международная конференция "Интеллектуальный анализ информации ИАИ-2005". – К.: Просвіта, 2005. – С.247-255.
346. *Рогушина Ю.В., Ершова О.Л.* Технология определения информационной ценности факторов, влияющих на поддержку и принятие решений в социально-экономических системах // Экономико-математическое моделирование социально-экономических систем. Сб.научн.тр. – Киев: Ин-т кибернетики им. В.М.Глушкова АН Украины, 1994. – С.37-40.
347. *Рогушина Ю.В., Протасова Э.Н., Москалькова Н.М.* Оценка релевантности методов дискретизации данных в задаче индуктивного обобщения // УСиМ, № 1, 1999. – С.54-58.

348. *Саати Т., Керне К.* Аналитическое планирование. Организация систем. – М.: Радио и связь, 1991. – 224 с.
349. Свободно распространяемые оболочки экспертных систем. – http://inf.susu.ac.ru/~pollak/expert/commercial/Q5-1.htm.
350. Системы менеджмента знаний (новые информационные технологии управления организацией, основанные на накоплении и использовании интеллектуальных активов), 2003. – http://agents.umbc.edu/bookmarks/awbookmarks.html.
351. *Старчевский И.П., Гончарук А.И., Черников А.В.* Производство и применение биологических средств защиты растений от вредителей и болезней // Материалы Международной научно-практической конференции "Информационное обеспечение технологических линий и биофабрик для промышленной наработки энтомокультур". – Одесса, 1994. – С.14-15.
352. *Стерлинг Х., Змиевский А.* PHP. Руководство разработчика. – К.: Диасофт, 2001. – 384 с.
353. *Сэлтон Г.* Автоматическая обработка, хранение и поиск информации. – М.: Сов. радио, 1973. – 560 с.
354. *Таран Т.А. Шемаев В.Н.* Метод моделирования рефлексивного управления развитием ситуаций в социально-экономических системах // 5 Межд.конф."Интеллектуальный анализ информации ИАИ-2005". – К.: Просвіта, 2005. – С.265-276.
355. *Тарасов В.Б.* От многоагентных систем к интеллектуальным организациям: философия, психология, информатика. – М.: Эдиториал УРСС, 2002. – 352 с.
356. *Тихонов В.* Архитектура метапоисковых систем. – http://www.citforum.ru/internet/search/metaping.html.
357. *Томсон Л., Веллинг Л.* Разработка Web-приложений на PHP и MySQL. – К.: Диасофт, 2001. – 672 с.
358. *Финн В.К.* О машинно-ориентированной формализации правдоподобных рассуждений в стиле Ф.Бэкана-Д.С.Милля // Семиотика и информатика. – 1983 – Вып.20. – С.24-41.
359. *Хан У., Мани И.* Системы автоматического реферирования // Открытые системы, N 12, 2000. – С.67-73.
360. *Хинтикка Я.* Виды модальностей // Семантика модальных и интерсиональных логик. – М.: Прогресс, 1981.– С.41-69.
361. Широков В.А. Феноменологія лексикографічних систем. – К.: Наук.думка, 2004. – 327 с.

Предметний покажчик

Список скорочень

AAT	Art and Architecture Tesaurus
ACC	Agent Communication Channel
ACL	Agent Communication Language
ACL	Agent Communication Language
ADL	Agent Definition Language
AMS	Agent Management System
AP	Agent Platform
API	Application Programming Interface
APSM	Agent Platform Security Manager
ARB	Agent Resource Broker
CORBA	Common Object Request Broker Architecture
CPS	Cooperative Problem Solving
D	Descriptors
DAML	DARPA Agent Markup Language
DAML-S	DAML Semantic Markup for Web Services
DCOM	Distributed COM
DDL	Description Definition Language
DOM	Document Object Model
DS	Description Schemes
DTD	Document type definition
FIPA	Foundation for Intelligent Physical Agents
HAP	Home Agent Platform
HTML	HyperText Markup Language
HTTP	Hypertext Transmission Protocol
ICL	Interagent Communication Language
IDL	Interface Definition Language
IIOP	Internet Inter-ORB Protocol
IPMT	Internal Platform Message Transport
IRE	Identifying Referring Expression
ISO	Міжнародна організація зі стандартизації
MCF	Meta Content Framework
MIME	Multipurpose Internet Mail Extensions
MPEG	Moving Picture Experts Group
OIL	The Ontology Inference Layer
OMG	Object Management Group
ORB	Object Request Broker
OWL	Web Ontology Language
P3P	Platform for Privacy Preferences

PGML	Precision Graphics Markup Language
PICS	Platform for Internet Content Selection
PNG	Portable Network Graphics
RADL	Reticular Agent Definition
RAMM	Reticular Agent Mental Model
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RMI	Remote Method Invocation, an inter-process communication method embodied in Java
SGML	Standard Generalized Markup Language
SMIL	Synchronized Multimedia Integration Language
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SVG	Scalable Vector Graphics
TCP / IP	Transmission Control Protocol / Internet Protocol
UDDI	Universal Description, Discovery and Integration
UDMA	User Dialogue Management Agent
UDMS	User Dialogue Management Service
UPA	User Personalization Agent
UPS	User Personalization Service
URI	Unified Resource Identifier
VML	Vector Markup Language
VRML	Virtual Realty Modelling Language
WSDL	Web Service Description Language
XMI	Metadata Interchange Format
XML	eXtensible Markup Language
XQL	XML Query Language
АН	Автоматизоване навчання
АОП	Агентно-орієнтоване програмування
БД	База даних
БЗ	База знань
ІАС	Інтелектуальна автоматизована система
ІЗ	Інженер зі знань
ІПА	Інформаційно-пошуковий агент
ІПС	Інформаційно-пошукова система
ІР	Інформаційні ресурси
ІС	Інформаційна система
ІТ	Інформаційна технологія
МАІПС	Мультиагентна інформаційно-пошукова система

МАС	Мультиагентна система
МПЗ	Модель подання знань
МПС	Метапошукова система
ПА	Програмні агенти
ПМ	Пошукова машина
ПрО	Предметна область
РШІ	Розподілений штучний інтелект
СБЗ	Система, що базується на знаннях
ШІ	Штучний інтелект

Наукове видання

ПЛЕСКАЧ Валентина Леонідівна
РОГУШИНА Юлія Віталіївна

АГЕНТНІ ТЕХНОЛОГІЇ

Монографія

Київський національний
торговельно-економічний університет, 2005

Редактор Л.В.Твердова
Комп'ютерна верстка Ю.В.Рогушина
Дизайн обкладинки Т.В. Матвієнко

Підп. до друку 20.09.2004. Формат 60х84х16. Папір. письм.
Ризографія. Ум.-друк. арк. 13,94. Ум. фарбо. відб. 12,96.
Обл.-вид.арк. 13,08. Тираж 300 пр. Зам. 682

Центр підготовки навчально-методичних видань КНТЕУ
02156, Київ-56, вул. Кіото, 19

Свідоцтво про державну реєстрацію сер. ДК, № 359 від 14.03.2001 р.

Довідка про авторів