

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт
з дисципліни
“Агентно-орієнтоване програмування”
для студентів спеціальності
122 “Комп’ютерні науки та інформаційні технології”
(денної форми навчання)

2016

Методичні вказівки до виконання лабораторних робіт з дисципліни “Агентно-орієнтоване програмування” для студентів спеціальності 122 “Комп’ютерні науки та інформаційні технології” (денної форми навчання) / О.О. Олійник, А.О. Олійник, В.М. Льовкін. – Запоріжжя : ЗНТУ, 2016. – 61 с.

Автори: О.О. Олійник, к.т.н., доцент
А.О. Олійник, к.т.н., доцент
В.М. Льовкін, к.т.н., доцент

Рецензент: С.О. Субботін, д.т.н., професор

Відповідальний
за випуск: С.О. Субботін, д.т.н., професор

Затверджено
на засіданні кафедри
програмних засобів

Протокол № 1
від “16” серпня 2016 р.

ЗМІСТ

Вступ	5
1 Лабораторна робота № 1 BDI-платформа JADEX	6
1.1 Мета роботи	6
1.2 Короткі теоретичні відомості	6
1.2.1 Основні принципи роботи платформи JADEX	6
1.2.2 Налаштування роботи з платформою JADEX у Eclipse	7
1.2.3 Основи створення планів агентів	12
1.3 Завдання до роботи	13
1.4 Зміст звіту	13
1.5 Контрольні запитання	13
2 Лабораторна робота № 2 Використання планів, переконань та можливостей при створенні агентів	15
2.1 Мета роботи	15
2.2 Короткі теоретичні відомості	15
2.2.1 Плани	15
2.2.2 База переконань	18
2.2.3 Використання можливостей	21
2.3 Завдання до роботи	24
2.4 Зміст звіту	25
2.5 Контрольні запитання	25
3 Лабораторна робота № 3 Розроблення інтелектуальних агентів	26
3.1 Мета роботи	26
3.2 Короткі теоретичні відомості	26
3.3 Завдання до роботи	30
3.4 Зміст звіту	30
3.5 Контрольні запитання	31
4 Лабораторна робота № 4 Goal-орієнтоване програмування	32
4.1 Мета роботи	32
4.2 Короткі теоретичні відомості	32
4.3 Завдання до роботи	36
4.4 Зміст звіту	36
4.5 Контрольні запитання	37

5 Лабораторна робота № 5 Взаємодія з зовнішніми процесами

в JADEX.....	38
5.1 Мета роботи	38
5.2 Короткі теоретичні відомості	38
5.3 Завдання до роботи.....	42
5.4 Зміст звіту.....	42
5.5 Контрольні запитання	43

6 Лабораторна робота № 6 Розроблення мультиагентної системи.....

системи.....	44
6.1 Мета роботи	44
6.2 Короткі теоретичні відомості	44
6.3 Завдання до роботи.....	48
6.4 Зміст звіту.....	48
6.5 Контрольні запитання	49

7 Лабораторна робота № 7 Розроблення web-агента

7.1 Мета роботи	50
7.2 Короткі теоретичні відомості	50
7.2.1 Основні поняття про програмні агенти	50
7.2.2 Властивості програмних агентів	50
7.2.3 Класифікація агентів	52
7.2.4 Web-агент	53
7.3 Завдання до роботи.....	56
7.4 Зміст звіту.....	58
7.5 Контрольні запитання	59

Література.....

60

ВСТУП

Дане видання призначене для вивчення та практичного засвоєння студентами денної форми навчання основ агентно-орієнтованого програмування.

Відповідно до графіка студенти перед виконанням лабораторної роботи повинні ознайомитися з конспектом лекцій та рекомендованою літературою. Дані методичні вказівки містять тільки основні, базові теоретичні відомості, необхідні для виконання лабораторних робіт, тому для виконання лабораторної роботи та при підготовці до її захисту необхідно ознайомитись з конспектом лекцій та опрацювати весь необхідний матеріал, наведений в переліку рекомендованої літератури, використовуючи також статті в інтернет-виданнях та актуальних наукових журналах з програмної інженерії.

Для одержання заліку з кожної роботи студент повинен у відповідності зі всіма наведеними вимогами розробити програмне забезпечення та оформити звіт, після чого продемонструвати на комп'ютері розроблене програмне забезпечення з виконанням всіх запропонованих викладачем тестів.

Звіт виконують на білому папері формату А4 (210 x 297 мм). Текст розміщують тільки з однієї сторони листа. Поля сторінки з усіх боків – 20 мм. Аркуші вміщують у канцелярський файл.

Усі завдання повинні виконуватись студентами індивідуально і не містити ознак плагіату як в оформленому звіті так і в розробленому програмному забезпеченні.

Під час співбесіди при захисті лабораторної роботи студент повинен виявити знання щодо мети роботи, теоретичного матеріалу, методів виконання кожного етапу роботи, змісту основних розділів звіту з демонстрацією результатів на конкретних прикладах, практичних прийомів використання теоретичного матеріалу в розробленому програмному забезпеченні. Студент повинен вміти обґрунтувати всі прийняті ним рішення та правильно аналізувати і використовувати на практиці отримані результати.

Для базової самоперевірки при підготовці до виконання і захисту роботи студент повинен відповісти на контрольні запитання, наведені наприкінці опису відповідної роботи.

1 ЛАБОРАТОРНА РОБОТА № 1

BDI-ПЛАТФОРМА JADEX

1.1 Мета роботи

1.1.1 Вивчити основні принципи моделі BDI.

1.1.2 Навчитися розробляти програмних агентів за допомогою засобів платформи JADEX.

1.2 Короткі теоретичні відомості

1.2.1 Основні принципи роботи платформи JADEX

JADEX представляє собою BDI-розширення мультіагентної платформи. Це середовище передбачає гібридну реактивно-деліберативну архітектуру, в якій агент розглядається як «чорна скриня», що приймає і відправляє повідомлення. Ґрунтуючись на результатах обробки повідомлень, внутрішніх і зовнішніх подій, деліберативний механізм приймає рішення про перехід до нового плану дій або продовження старого. Чинний план може надсилати повідомлення іншим агентам, змінювати базу переконань, формувати нові цілі і викликати внутрішні події. Система використовує бібліотеку планів, що обробляються як Java-класи.

JADEX – це засноване на Java FIPA-сумісне агентне середовище, що дозволяє у відповідності з моделлю BDI розробляти цільові агенти. JADEX забезпечує базову структуру і набір інструментів розробки, що полегшує створення і тестування агентів. Java-основа полягає в тому, що проект JADEX націлений на максимально просте розроблення агентних систем. Для того щоб сприяти плавному переходу від традиційних розподілених систем до розроблення (багато-)агентних систем, повинні бути за можливості застосовані об'єктно-орієнтовані поняття і технології. JADEX дозволяє полегшити реалізацію агентів за допомогою засобів широко розповсюдженої мови програмування Java, використовуючи при цьому величезну кількість існуючих інструментів і бібліотек.

Платформа JADEX призначена для створення агентів, які взаємодіють з системою, до якої підключений агент, і змінюють події систем в залежності від планів, що містить агент, цілей і подань

об'єктів, якими був запрограмований агент. Щоб створити агента необхідно визначити в системі властивості агента. Дії агента описуються за допомогою уявлень, цілей і планів. Поміщаючи їх, в так звану бібліотеку XML файлу, створюється файл ADF (Agent Definition File) – файл визначення агента. У ADF-файлі визначається агент як об'єкт, описаний в класах java-файлів. Запуск агента визначається в планах, які викликаються на вимогу системи так званими пасивними планами.

Функціонування агента в системі відбувається за рахунок отримання від системи і відправлення змінених повідомлень в систему. Система подає на вхід агенту якесь повідомлення, після чого агент заносить його в так званий модуль подій, передаючи в блок обробки повідомлень і далі по ланцюжку в блок планів, де виконується аналіз події на приналежність до агента. Якщо повідомлення не належить агенту, то воно передається назад в систему не зміненим. Блок планів, оброблюючи події, порівнює їх з власною базою планів і за позитивного порівняння з планами починає оброблення повідомлення в базі знань, звідки воно надходить для порівняння з цілями агента для передачі в модуль подій. Пройшовши весь цикл порівняння і перетворення даних в установлені завдання, виконується передача в систему для подальшого функціонування.

Інтерпретатор складається зі списку інформації компонента основних планів розв'язування задачі – так звані заплановані метадії, орієнтовані на виконання агентом. Режим роботи приблизно виглядає так: агент вибирає метадію з файлу ADF і виконує її, коли виконуються попередні умови дії. Інакше дія просто пропускається. Виконання дії може призвести до подальших дій, які додані до списку ADF-файлу.

1.2.2 Налаштування роботи з платформою JADEX у Eclipse

Для підключення JADEX потрібно завантажити плагін Eclipse-JADEX-0.1.0.tar.bz2, який можна знайти, наприклад, за цим посиланням <https://code.google.com/p/plugin-eclipse-JADEX/downloads/detail?name=Eclipse-JADEX-0.1.0.tar.bz2>

Після завантаження необхідно розархівувати отриманий файл та скопіювати jar-файли у папку plugins встановленого продукту Eclipse (рис. 1.1).

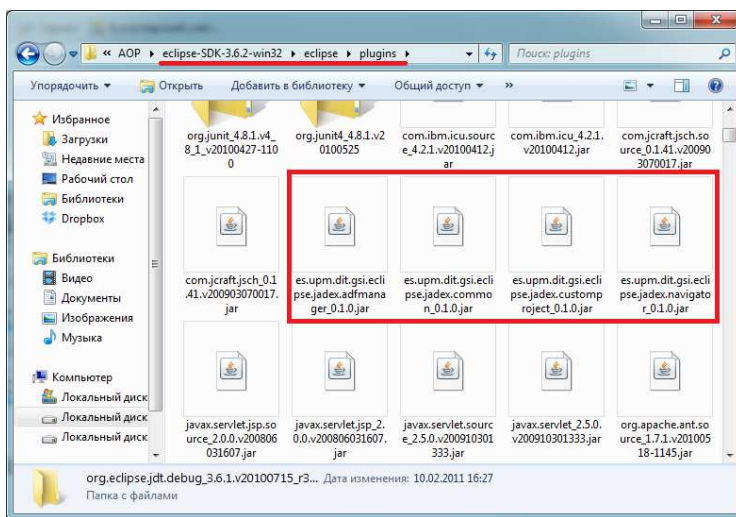


Рисунок 1.1 – JADEX плагін у Eclipse

Тепер потрібно відкрити Eclipse та змінити шлях директорії JADEX. Для цього зайти у меню Window → Preferences. У вікні знайти пункт Eclipse JADEX, обрати шлях до ресурсу JADEX та вказати версію (рис. 1.2).

Ресурс JADEX потрібно попередньо завантажити, наприклад, за посиланням <http://www.activecomponents.org/bin/view/Download/Overview>. Після обрання шляху до бібліотеки потрібно натиснути кнопки Apply та OK. Тепер плагін підключений, можна працювати з ним.

Для того щоб створити агента, потрібно обрати пункт меню File → New → Project. У діалоговому вікні обрати JADEX Project, як показано на рис. 1.3, та натиснути кнопку Next.

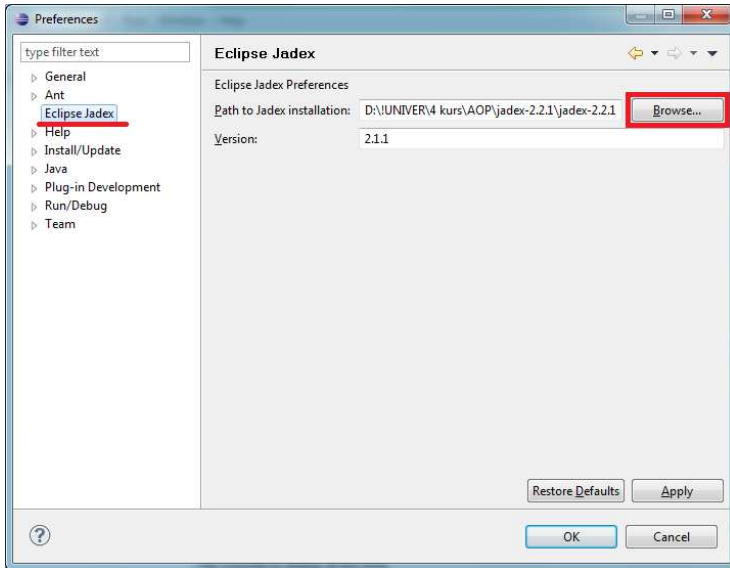


Рисунок 1.2 – Зміна шляху до ресурсу JADEx

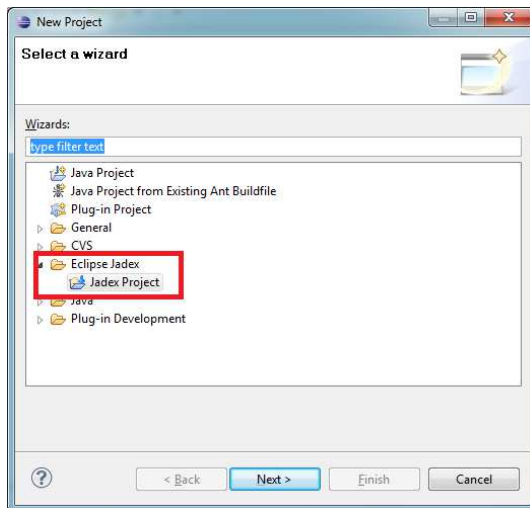


Рисунок 1.3 – Створення проекту JADEx

У наступному вікні вказати ім'я проекту та натиснути кнопку Finish. Далі зайти у меню File → New → Other та обрати JADEx Agent

Description File у вікні, що з'явиться (рис. 1.4), і натиснути Next. В наступному вікні обрати проект, який був створений раніше, дати ім'я агенту та пакету (рис. 1.5).

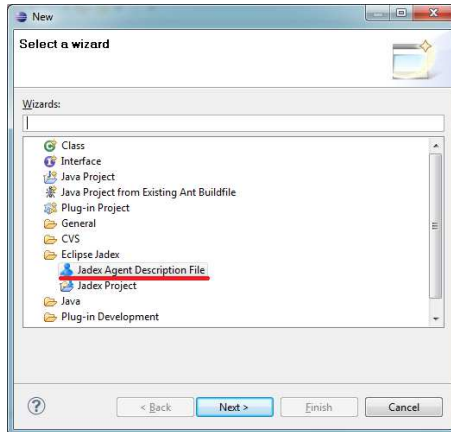


Рисунок 1.4 – Вікно створення файлу агента

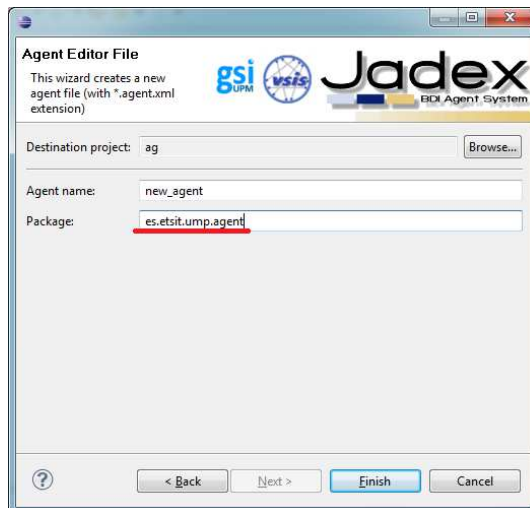


Рисунок 1.5 – Приклад імені та розгортання агента

Приклад створеного агента представлений на рис. 1.6. На панелі зліва можна побачити розгортання агента.

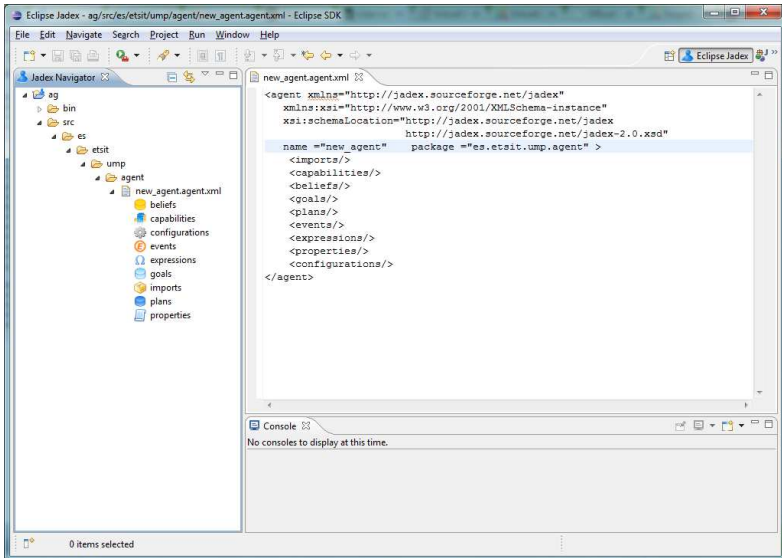


Рисунок 1.6 – Приклад створеного агента

Після того, як агент створений та реалізовано для нього всю необхідну функціональність, необхідно запустити його на виконання, для чого призначений пункт контекстного меню проекту Run As → Java Application.

Далі необхідно у вікні, що відкріється, знайти клас Starter з пакету JADEX.base.

Після цього на екрані з'явиться JADEX Control Center (рис. 1.7) – інтерфейс для зневадження та керування для платформи JADEX та компонентів, які на ній запускаються. Звідти можна обрати відповідний компонент, що представляє розробленого агента, та запустити його, після чого з'являться відповідні результати роботи агента у відповідності з логікою його роботи.

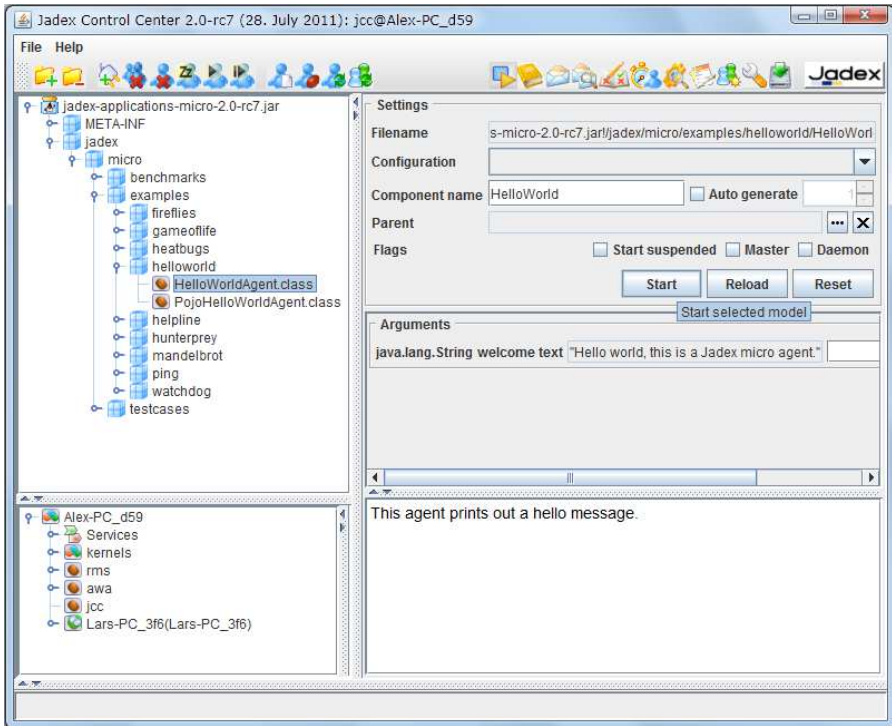


Рисунок 1.7 – Запуск компонента в JADEX Control Center

1.2.3 Основи створення планів агентів

Плани відіграють головну роль у JADEX, тому що вони інкапсулюють спосіб досягнення деякого стану.

Новий план створюється у проекті як звичайний java-файл.

Для опису плану спочатку необхідно імпортувати клас JADEX.bdi.runtime.Plan у java-файл. Обов'язково потрібно розширити створений клас, дописуючи до оголошення класу extends Plan:

Java

```
public class Name_plan extends Plan
```

У java-файлі можна описати методи, що будуть використовуватись у тілі класу.

1.3 Завдання до роботи

1.3.1 Ознайомитися з літературою та основними теоретичними відомостями, необхідними для виконання роботи.

1.3.2 Ознайомитися з бібліотекою JADEX.

1.3.3 Підключити бібліотеку JADEX до проекту, створеного у Eclipse.

1.3.4 Узгодити з викладачем індивідуальне завдання.

1.3.5 Створити агента у відповідності з індивідуальним завданням.

1.3.6 Виконати тестування та проаналізувати роботу створеного агента.

1.3.7 Оформити звіт з роботи.

1.3.8 Відповісти на контрольні запитання.

1.4 Зміст звіту

1.4.1 Мета роботи.

1.4.2 Текст програми (з коментарями), що реалізує роботу агента.

1.4.3 Результати тестування агента.

1.4.4 Копії екранних форм, що реалізують взаємодію користувача з програмним агентом.

1.4.5 Висновки, що містять відповіді на контрольні запитання, а також відображують результати виконання роботи та їх критичний аналіз.

1.5 Контрольні запитання

1.5.1 Що таке BDI?

1.5.2 Яку модель реалізує JADEX?

1.5.3 Які особливості створення агентів у JADEX?

1.5.4 Яким чином виконувати розроблення агентів із використанням бібліотеки JADEX?

1.5.5 Опишіть особливості запуску агентів, створених із використанням бібліотеки JADEX.

1.5.6 Які існують види агентів та моделей їх представлення?

1.5.7 Які особливості агентно-орієнтованого програмування?

1.5.8 Які засоби керування містить Java Control Center?

1.5.9 Що таке ADF-файл та яка його структура?

1.5.10 У чому полягає механізм взаємодії в межах платформи JADEX?

2 ЛАБОРАТОРНА РОБОТА № 2 ВИКОРИСТАННЯ ПЛАНІВ, ПЕРЕКОНАНЬ ТА МОЖЛИВОСТЕЙ ПРИ СТВОРЕННІ АГЕНТІВ

2.1 Мета роботи

2.1.1 Вивчити основні принципи опису планів, переконань та можливостей агентів.

2.1.2 Навчитися реалізовувати агентів за допомогою засобів JADEX на основі опису їх планів, переконань та можливостей.

2.2 Короткі теоретичні відомості

2.2.1 Плани

На відміну від інших відомих PRS-подібних систем, JADEX має дві форми планів.

Миттєвий план – це план, який створюється на час запуску агента. Це досягається шляхом створення верхнього рівня виконання мети цього плану агента. Миттєвий план – простий спосіб реагувати на запити послідовного обслуговування, без необхідності синхронізувати різні екземпляри плану за тією ж схемою. Тому миттєвий план може налаштувати свою особисту чергу очікування подій і отримувати події, навіть якщо вони працюють на даний момент.

Пасивний план є звичайною версією плану, який можна знайти у всіх інших PRS-системах. Для такого плану фільтр повинен бути вказаний у файлі визначення агента, щоб дозволити агенту знати, які види подій цей план може обробити. Коли подія надходить до агента, диспетчер створює так відповідний список плану APL (Applicable Plan List), в якому всі плани з відповідними фільтрами і один вибраний і створений для виконання. Пасивні плани є гарним вибором, коли необхідно паралельне виконання одного виду завдань або це найпростіше не заважає.

Часто план виконує деякі дії, а потім повинен з'єднатися з іншим агентом і повинен чекати відповіді. Тому план може використовувати метод `waitFor (filter [, timeout])`, який дозволяє

планам задати фільтр, якому відповідає подія (або події) реагування на план (можливо, зазначений з деякими тайм-аутом).

Зверніть увагу, що метод `action()` виконується після оголошення `WaitFor()`, це відбувається через однопоточну модель виконання, успадковану від JADE.

Існують різні фільтри, які можуть реагувати на різні типи подій. Щоб дочекатися листа у відповідь досить створити JADE `MessageTemplate` і упакувати його в `MessageFilter`.

Розглянемо деякі особливості планів на прикладі створення плану англійсько-німецького перекладача.

Спочатку необхідно створити новий файл з ім'ям `EnglishGermanTranslationPlanP1.java`, відповідальний за основний переклад слова, що:

- розширюється `JADEX.Plan` класом;
- імпортує `JADEX` класи;
- включає в себе аргумент конструктора, який виводить текст "Created:" + `this` і встановлює постійний (черга очікування) фільтр для плану. Тому необхідно додати статичний метод `IFilter getEventFilter()`, який повертає фільтр, який може бути встановлений як постійний фільтр в конструктор з командою `addPermanentFilter(getEventFilter())`. Команда `Additionally WaitFor(getEventFilter())` повинна бути викликана після того, як встановлений постійний фільтр. Це вказує на те, що план зацікавлений в особливого роду події, що визначена фільтром подій.

`GetEventFilter()` повинен створити `MessageFilter`, який відповідає всім повідомленням з перформативною `query_ref` за допомогою відповідного JADE `MessageTemplate`.

Java

```
MessageTemplate mt = MessageTemplate.MatchPerformative
(ACLMessage.QUERY_REF);
MessageFilter mfilter = new MessageFilter (mt);
```

Реалізуємо action (Event event) метод. Замість виконання запиту бази даних будемо використовувати просту геш-таблицю для пошуку слів. Створення та ініціалізацію цього слова таблиці з декількома парами слів уже можна зробити в конструкторі. У результаті план повинен друкувати "Переклад з англійської на німецьку:" + Eword + "-" + gword або "Вибачте, немає у базі даних:" + eword. Щоб отримати ACLMessage із запитом події використовують ((MessageEvent) event). GetMessage().

Тепер можна витягти вміст повідомлення як зазвичай (за допомогою (using <ACLMessage>. GetContent())). Наприкінці методу action() має бути зазначено, що план тепер готовий до обробки подальших запитів. Тому додамо команду waitFor (getEventFilter ())десь в кінці плану.

Додамо план в агента, визначаючи його у файлі опису агента, тобто в ADF-файлі.

Оголосимо використання миттєвих планів, додавши: instant_plans = egtrans

Оголосимо відповідність між назвою плану і плану створення, додавши рядок: egtrans = EnglishGermanTranslationPlanP1()

Якщо було створено план агента в пакеті, необхідно обов'язково повністю визначити ім'я класу або використовувати оператор імпорту у властивості файлу. Наприклад, це може виглядати так: imports = examples.tutorial [, other_packages].

ADF

```
<plans>
  <!--Опис плану. -->
  <plan name="Plane_name">
    <body impl="Name_of_the_Plan_class"/>
  </plan>
</plans>
```

Обов'язково потрібно оголосити план у розділі конфігурації ADF-файлу.

ADF

```
<configurations>
  <configuration name="default">
    <plans>
      <!--Запускаємо потрібний план, що
був ініціалізований раніше. -->
      <initialplan ref=" Plane_name "/>
    </plans>
  </configuration>
</configurations>
```

2.2.2 База переконань

База переконань агента представляє його знання про світ. База переконань в деякому роді схожа на просте зберігання даних, що дозволяє очистити зв'язки між різними планами за допомогою загальних переконань.

На відміну від більшості BDI систем PRS-виду, JADEX дозволяє зберігати довільні судження як переконання в базі переконань.

У JADEX між двома видами переконань є відмінності. З одного боку, є переконання, які дозволяють користувачу зберігати рівно один факт, а з іншого боку підтримуються набори переконань, що дозволяють зберігати набір фактів. Використання переконання в якості первинних потужностей для зберігання планів настійливо рекомендується, тому що від такого використання є декілька переваг для користувачів.

Підтримується декларативна мова запитів, якщо необхідно отримати частину збережених даних. Крім того можна контролювати одиночні переконання відносно до свого становища і викликати події в разі, якщо відповідна умова досягнута. Це дозволяє викликати деякі дії, коли, наприклад, факт набору переконань доданий або переконання змінене. Крім того, можна чекати деякого складного

виразу, який відноситься до декількох переконань, щоб бути виконаним.

Особливості використання переконань розглянемо на прикладі агента англо-німецького перекладача.

Ми використовуємо семантики у форматі запиту, який виглядає наступним чином: дія-мова(и)-зміст. Щоб перекласти слово, ми повинні відправити запит у вигляді: перекласти english_german слово (translate english_german word). Щоб додати нову пару слів в базу даних, ми повинні відправити запит у форматі: додати english_german eword gword (add english_german eword gword).

Змінимо існуючий план, щоб підтвердити формат запиту і ввести новий план для додавання пари слів.

Створимо новий план AddGermanWordPlanB1 як пасивний план, який обробляє запити доповнення нових пар слів. План повинен перевіряти свій метод дії, якщо формат запиту правильний (за допомогою StringTokenizer). Якщо все нормально, повинна вийти геш-таблиця, що містить пари слів через:

Java

```
Hashtable words = (Hashtable) getAgent().GetBeliefbase().getFact(
    (EnglishGermanTranslationPlanB1.BELIEF_ENG_GER_WORDTABLE)
```

Тепер план повинен перевірити, чи міститься англійське слово в геш-таблиці (за допомогою <hashtable>. ContainsKey (eword)) і якщо він не міститься, слід додати (за допомогою <hashtable>. Put (eword, gword)).

Оголосимо план, зіставляючи з виразом створення і виразом фільтра в файлі ADF.

Реалізуємо getEventFilter () для цього плану і нехай він реагує на повідомлення, які починаються з "add english_german". Потім необхідно створити

Java

```
MessageTemplate mt = new MessageTemplate
(neMatchStartContentLiteral ("add english_german"))
```

і упакувати `MessageTemplate` в `MessageFilter`.

Змінімо план `EnglishGermanTranslationPlanB1` так, щоб він зберігав готове слово таблиці як одне переконання в базі думок. Тому необхідно створити думку і ініціалізувати її зі словом у таблиці. Належне місце для створення думок вже є в ADF. Додайте рядок оголошення думки: `beliefs = egwords` і відображення цього переконання `egwords = Hashtable ()` і лінії, для визначення імені переконання

Java

```
egwords_name=EnglishGermanTranslationPlanB1.
BELIEF_ENG_GER_WORDTABLE
```

Для того, щоб використовувати клас `Hashtable`, ви повинні додати пакет `java.util` в рядок імпорту. В імпорт потрібно додати через кому список пакетів, які будуть використовуватися для парсування ADF. Синтаксис: `imports = <package1>, <package2>, ...` (Наприклад, `imports = java.util`). Зверніть увагу, що тільки пакети не поодиноких класів можуть бути імпортовані.

Визначте константу, яка визначає переконання слова таблиці в перекладі плану:

Java

```
public static final String BELIEF_ENG_GER_WORDTABLE
= "eng_ger_wordtable";
```

Далі необхідно змінити `EnglishGermanTranslationPlanB1` так, щоб він перевіряв, чи введений новий формат запиту за допомогою `StringTokenizer`.

Щоб додати новий факт в існуючий набір думок можна використовувати метод:

Java

```
<beliefbase>.addFact(EnglishGermanTranslationPlanB2.  
BELIEF_ENG_GER_WORDTABLE, new Tuple (eword,  
gword))
```

2.2.3 Використання можливостей

Різним агентам часто доводиться використовувати ті ж або аналогічні функціональні можливості, які включають більше, ніж просто план поведінки. Часто приватні або загальні переконання і цілі є частиною загальної функціональності одного агента. Ці одиниці функціональності співставні з модулем концепції об'єктно-орієнтованої парадигми, але володіють дуже різними властивостями завдяки використанню менталістських понять. З цих причин можливість концепцій була введена спочатку і посилюється, що дозволяє упаковувати підмножину переконання планів і цілей в агентських модулях і використовувати цей модуль там, де це необхідно.

Структура можливостей агента утворює дерево. Можливість успадковувати (батька) може містити довільне число субможливостей. Усі елементи мають можливість за замовчуванням приватної видимості (`private visibility`) і повинні бути явно надані для використання в зв'язковій можливості.

Розглянемо деякі особливості можливостей на прикладі створення плану англо-німецького перекладача.

Оголосить дві змінні на початку плану:

Java

```
String reply;
String content;
```

Далі потрібно встановити значення обох змінних по відношенню до правильності перекладу. У разі успіху набір (за умови, що gword і ework змінні для англійського та німецького слова відповідно) дорівнює:

Java

```
reply = "inform";
content = gword;
```

А в разі помилки:

Java

```
reply = "failure";
content = "Sorry, word could not be translated:" + ework;
```

Надіслати відповідь, що викликається в кінці обробки подій:

Java

```
IMessageEvent      replymsg      =
getEventbase().createReply((IMessageEvent)getReason(),
reply);
replymsg.getParameter(SFipa.CONTENT).setValue(content);
sendMessage(replymsg);
```

Додамо нові типи подій повідомлень "inform" і "failure" в ADF-файл:

ADF

```

<events>
  ...
  <messageevent name="inform" direction="send" type="fipa">
    <parameter      name="performative"      class="String"
direction="fixed">
      <value>SFipa.INFORM</value>
    </parameter>
  </messageevent>

  <messageevent name="failure" direction="send" type="fipa">
    <parameter      name="performative"      class="String"
direction="fixed">
      <value>SFipa.FAILURE</value>
    </parameter>
  </messageevent>
</events>

```

Створимо нову можливість ADF. Створимо новий файл TranslationD1.capability.xml зі скелетом коду в наступному фрагменті.

ADF

```

<capability xmlns = "http://JADEX.sourceforge.net/JADEX-bdi"
  xmlns: xsi = "http://www.w3.org/2001/XMLSchema-
instance"
  xsi:          schemaLocation          =
"http://JADEX.sourceforge.net/JADEX-bdi
             http://JADEX.sourceforge.net/JADEX-bdi-
2.0.xsd "
  name = "TranslationD1"
  package = "JADEX.bdi.tutorial">
  ...
</ capability>

```

Тепер необхідно скопіювати визначення імпорту, планів, переконань, подій (у тому числі ті, що визначені зверху) і вирази від раніше створеного TranslationC2.agent.xml в цей файл.

Далі необхідно змінити агент ADF (TranslationD1.agent.xml) шляхом видалення всіх оголошень планів і думок. Замість цього, вставити новий розділ з використання нових можливостей.

ADF

```
<capabilities>
  <capability name="transcap" file="TranslationD1"/>
</capabilities>
```

Відзначимо, що тут використовується ім'я типу, але також можуть бути використані абсолютні та відносні шляхи до файлу XML.

2.3 Завдання до роботи

2.3.1 Ознайомитися з літературою та основними теоретичними відомостями, необхідними для виконання роботи.

2.3.2 Ознайомитися з планами, переконаннями, можливостями агентів.

2.3.3 Підключити бібліотеку JADEX до проекту, створеного у Eclipse.

2.3.4 Узгодити з викладачем індивідуальне завдання.

2.3.5 Визначити та описати переконання агента та його плани дій у вигляді тексту.

2.3.6 Реалізувати агента та його переконання у відповідності з індивідуальним завданням. Реалізація завдання має ґрунтуватися на наступних особливостях: можливість створення різноманітних планів дій, виконання дій на базі певних переконань, можливість взаємодії із середовищем.

2.3.7 Вивести переконання у консолі. Створити новий план, що буде виводити попередньо створену базу переконань.

2.3.8 Виконати тестування та проаналізувати роботу створеного агенту.

2.3.9 Оформити звіт з роботи.

2.3.10 Відповіді на контрольні запитання.

2.4 Зміст звіту

2.4.1 Мета роботи.

2.4.2 Текст програми (з коментарями), що реалізує роботу агента.

2.4.3 Результати тестування агента.

2.4.4 Копії екранних форм, що реалізують взаємодію користувача з програмним агентом.

2.4.5 Висновки, що містять відповіді на контрольні запитання, а також відображують результати виконання роботи та їх критичний аналіз.

2.5 Контрольні запитання

2.5.1 Що таке план?

2.5.2 Для чого потрібні плани?

2.5.3 Які існують види планів? Чим вони відрізняються?

2.5.4 Яку бібліотеку необхідно підключити для роботи з планами?

2.5.5 Які методи дозволяють керувати планами агентів?

2.5.6 Що таке переконання? Для чого вони служать?

2.5.7 Які існують види переконань у JADEX?

2.5.8 Яким чином можна витягти переконання агента?

2.5.9 Де відбувається опис агента?

2.5.10 Яким чином змінити переконання агента?

2.5.11 Що таке можливості? Для чого вони служать?

3 ЛАБОРАТОРНА РОБОТА № 3 РОЗРОБЛЕННЯ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ

3.1 Мета роботи

Навчитися реалізовувати інтелектуальних агентів у середовищі JADEX.

3.2 Короткі теоретичні відомості

На даний момент виділяють такі властивості, якими повинні володіти інтелектуальні агенти (IA):

1) автономність (autonomy, autonomous functioning) – здатність до самостійного формування цілей і функціонування з самоконтролем своїх дій і внутрішнього стану;

2) суспільна поведінка (social ability, social behavior) – здатність узгодити свою поведінку з поведінкою інших агентів в умовах певного середовища і правил поведінки шляхом обміну повідомленнями мовою комунікації;

3) реактивність (reactivity) – здатність адаптуватися до стану зовнішнього середовища (середовища функціонування і безлічі інших агентів) і своєчасно реагувати на зміни, що відбуваються;

4) активність (pro-activity) – здатність проявляти ініціативу, тобто самостійно генерувати цілі та діяти раціонально для їх досягнення, а не лише пасивно реагувати на зовнішні події;

5) базові знання (basic knowledge) – постійна частина знань агента про себе, про середовище, а також постійні знання про інших агентів, які не змінюються в рамках життєвого циклу агента;

6) переконання (beliefs) – змінна частина знань агента про середовище та інших агентів, яка може змінюватися в часі, але агент може про це не знати і продовжувати використовувати їх для своїх цілей;

7) бажання (desires) – стан та / або ситуації, досягнення яких є бажаним і важливим для агента, однак які можуть бути суперечливими і не всі будуть досягнуті;

8) цілі (goals) – сукупність станів, на досягнення яких спрямована поточна поведінка агента;

9) наміри (intentions) – це дії, які агент зобов'язаний зробити в силу своїх зобов'язань по відношенню до інших агентів, або те, що випливає з його бажань (тобто несуперечлива підмножина бажань, вибрана з тих чи інших причин і сумісна з прийнятими на себе зобов'язаннями);

10) зобов'язання (commitments) – завдання, які бере на себе агент на прохання та / або дорученням інших агентів.

Для адаптивних агентів (АА) постулюються такі властивості як адаптивність (здатність діяти відповідно і узгоджено зі змінами в середовищі), робастність (малі зміни у властивостях середовища не повинні призводити до неможливості функціонування АА), наявність тактики (здатність встановлювати множинні цілі і незалежно від обставин вирішувати, які цілі активно переслідувати), гнучкість (здатність виконувати широке коло завдань).

У коді агентів, написаних мовою Java, можна використовувати засоби інтелектуального аналізу даних багатьма способами. Наприклад, можна скористатися класами програмного середовища Weka для:

- weka.core.Instances – доступу до даних;
- weka.filters – попереднього опрацювання даних;
- weka.classifiers/weka.clusterers – класифікації або кластеризації на основі опрацьованих даних;
- weka.classifiers.Evaluation/weka.clusterers.ClusterEvaluation – оцінювання того, на скільки якісним виявився класифікатор або кластеризатор;
- weka.attributeSelection – вилучення неінформативних атрибутів з даних.

Приклад читання даних з файлу ARFF:

Java

```
import weka.core.Instances;
import java.io.BufferedReader;
import java.io.FileReader;
...
BufferedReader reader = new BufferedReader(
    new FileReader("/some/where/data.arff"));
Instances data = new Instances(reader);
```

Java

```
reader.close();
data.setClassIndex(data.numAttributes() - 1);
```

Розглянемо приклад попереднього опрацювання даних зі створенням та застосуванням фільтра:

Java

```
import weka.core.Instances;
import weka.filters.Filter;
import weka.filters.unsupervised.attribute.Remove;
...
String[] options = new String[2];
options[0] = "-R";
options[1] = "1";
Remove remove = new Remove();
remove.setOptions(options);
remove.setInputFormat(data);
Instances newData = Filter.useFilter(data, remove);
```

Приклад побудови класифікатора:

Java

```
import weka.classifiers.trees.J48;
...
String[] options = new String[1];
options[0] = "-U";
J48 tree = new J48();
tree.setOptions(options);
tree.buildClassifier(data);
```

Дані мають бути перетворені в об'єкт Instance, який розпізнає Weka. Об'єкт Instance містить набір значень, з якими класифікатор вміє працювати. Залежності між метаданими ознак, такими як імена, та значеннями не зберігаються в об'єкті Instance. Деталі ознак, зокрема тип, зберігаються в окремому об'єкті Instances, який може містити множину об'єктів Instance.

Приклад побудови кластеризатора:

Java

```
import weka.clusterers.EM;
...
String[] options = new String[2];
options[0] = "-I";
options[1] = "100";
EM clusterer = new EM();
clusterer.setOptions(options);
clusterer.buildClusterer(data);
```

Ще одним способом є використання засобів пакета Matlab.

Matlabcontrol – це Java API, який дозволяє викликати MATLAB з Java.

Базовий шаблон взаємодії складається зі створення фабрики та проксі, який використовується для комунікації з MATLAB:

Java

```
MatlabProxyFactory factory = new MatlabProxyFactory();
MatlabProxy proxy = factory.getProxy();
proxy.eval("disp('hello world')");
proxy.setVariable("a", 5);
proxy.eval("a = a + 6");
Object result = proxy.getVariable("a");
System.out.println("Result: " + result);
proxy.eval("array = randn(4,3,2)");
proxy.eval("disp(['entry: ' num2str(array(3, 2, 1))])");
```

Java

```

MatlabTypeConverter processor = new MatlabTypeConverter
(proxy);
MatlabNumericArray array = processor.getNumericArray
("array");
System.out.println("entry: " + array.getRealValue(2, 1, 0));
double[][][] javaArray = array.getRealArray3D();
System.out.println("entry: " + javaArray[2][1][0]);
...
proxy.disconnect();

```

3.3 Завдання до роботи

3.3.1 Ознайомитися з літературою та основними теоретичними відомостями, необхідними для виконання роботи.

3.3.2 Узгодити з викладачем індивідуальне завдання на основі розширення агента з лабораторної роботи № 2 інтелектуальними засобами. Самостійно розширити завдання: запропонувати використання ще одного методу інтелектуального аналізу даних для проблеми, що розглядається.

3.3.3 Із використанням бібліотеки JADDEX реалізувати агента, що буде розв'язувати поставлене завдання.

3.3.4 Виконати тестування та проаналізувати роботу створеного агента.

3.3.5 Оформити звіт з роботи.

3.3.6 Відповісти на контрольні запитання.

3.4 Зміст звіту

3.4.1 Мета роботи.

3.4.2 Опис переконань агентів, планів дій, цілей агентів.

3.4.3 Текст розробленого програмного забезпечення з коментарями.

3.4.4 Результати тестування агента.

3.4.5 Результати роботи програмного забезпечення.

3.4.6 Висновки, що відображають особисто отримані результати виконання роботи та їх критичний аналіз.

3.5 Контрольні запитання

3.5.1 Які властивості повинні мати сучасні інтелектуальні агенти?

3.5.2 Що таке переконання агента типу BDI?

3.5.3 Що повинен обов'язково містити будь-який агент?

3.5.4 Чим відрізняються intentions та commitments?

3.5.5 Яким чином використати інтелектуальні методи аналізу даних з середовища Weka для реалізації агента?

3.5.6 Яким чином можна використати засоби пакету Matlab для реалізації інтелектуального агента?

3.5.7 Які методи кластеризації та яким чином можна використати для створення агента?

3.5.8 Яким чином можна виконати попереднє опрацювання даних у процесі роботи агента?

3.5.9 Які методи класифікації та яким чином можна використати для створення агента?

3.5.10 Які засоби відбору атрибутів може використовувати інтелектуальний агент?

4 ЛАБОРАТОРНА РОБОТА № 4

GOAL-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

4.1 Мета роботи

Вивчити принципи goal-орієнтованого програмування та навчитися використовувати їх у процесі створення агентів.

4.2 Короткі теоретичні відомості

Цілеспрямована поведінка є одним із ключових понять агент-орієнтованої парадигми. Це означає, що агент або план здійснює певні цілі і може використовувати всі можливості для досягнення своєї мети.

Агент буде випробовувати всі свої плани, поки його мета не стане успішною, у іншому випадку він не має можливості дістатися до будь-якої іншої мети.

Для адаптивних агентів (АА) постулюються такі властивості як адаптивність (здатність діяти відповідно і узгоджено зі змінами в середовищі), робастність (малі зміни у властивостях середовища не повинні призводити до неможливості функціонування АА), наявність тактики (здатність встановлювати множинні цілі і незалежно від обставин вирішувати, які цілі активно переслідувати), гнучкість (здатність виконувати широке коло завдань).

Поняття «агент», що розглядається в якості потужного прикладу розробки програмного забезпечення, дуже корисне у відношенні складності нинішніх програмних систем. Воно дозволяє розглянути систему як сформовану з самостійних взаємодіючих об'єктів (сутностей, модулів), які переслідують свої власні цілі і діють розумним чином. Отже, внутрішній стан і процес ухвалення рішення агентів моделюється в інтуїтивній формі, дотримуючись принципу інтелектуальних відносин. Цілеспрямованість означає, що розробник може визначити більш абстрактні цілі для агентів і тим самим забезпечити певний ступінь гнучкості підходів до їх досягнення. Модель BDI, заснована на інтелектуальних відносинах, спочатку представлялася як філософська модель для створення розумних (людських) агентів. Проте пізніше вона була адаптована і перетворена

в модель для програмних агентів, засновану на ідеології думок, цілей і планів. JADEX об'єднує цю модель в агентів Jade шляхом подання «думок», «цілей» і «планів» об'єктами першого класу, які можуть бути створені всередині агента і якими можна маніпулювати.

У JADEX в агентів є думки, здатні бути будь-яким типом Java-об'єкта і зберігатися в базі думок. Цілі – це явні чи неявні описи станів, які потрібно досягти. Щоб досягти своїх цілей, агент виконує плани, які є процедурними засобами, закодованими в Java.

Цілі для IA описуються в xml-файлі, який визначає їх для досягнення, тобто виконання заданої суті агента. Також формуються плани і знання (факти), які агент буде використовувати для досягнення і виконання своїх цілей.

Цілі в JADEX представляють собою об'єкти з деякими атрибутами. Цільовий стан цілі, що досягається, може бути явно представлено виразом, який перевіряє, чи досягнута ціль.

JADEX підтримує 4 види цілей:

- виконання;
- досягнення;
- запиту;
- підтримки.

Ціль виконання стверджує, що має бути виконане, але не обов'язково призведе до певного результату.

Ціль досягнення описує абстрактний цільовий стан, який необхідно досягти, без визначення як це зробити. Таким чином, агент може випробувати різні альтернативи, для досягнення мети.

Ціль запиту представляє необхідність в інформації. Якщо інформація на поточний момент не доступна, вибираються і виконуються плани, які допомагають її зібрати.

Ціль підтримки специфікує стан, який має зберігатися, і може викликатися після того, як буде виконана. Це сама абстрактна мета в JADEX.

Цілі становлять мотиваційну установку агента, яка є рушійною силою його дій. Тому уявлення і обробка цілей – це одна з головних функцій JADEX.

Мета-ціль може бути використана в процесі вибору плану для винесення висновку про події та відповідні плани.

У JADEX цілі сильно типізовані. Усі типи цілей можуть бути ідентифіковані через ім'я. Усі параметри цілі повинні бути декларовані в XML.

Goal-орієнтоване програмування представляє собою один з основних принципів агентно-орієнтованого програмування. Це означає, що агент, виходячи з певних реальних станів та особливостей, направляє свою діяльність на виконання певної мети (goal). Агент намагається виконувати кожен із своїх планів дій до тих пір, поки ціль не буде досягнута. Крім того, кожна ціль може розбиватися на цілі нижнього рівня (subgoal).

Щоб мати другий план для перекладу з англійської на французьку, введемо `ProcessTranslationRequestPlan`, який отримує всі вхідні запити на переклад і використовує підзадачі, викликані для досягнення мети.

Для виділення цілі та для розбивання на цілі нижнього рівня можна використовувати наступні засоби:

Java

```
IGoal goal = createGoal("translate");
goal.getParameter("direction").setValue(dir);
goal.getParameter("word").setValue(word);
try {
    dispatchSubgoalAndWait(goal);
    getLogger().info("Translated from "+goal.getName()+" "+
        word+" - "+goal.getParameter("result").getValue());
}
catch(GoalFailureException e) {
    getLogger().info("Word is not in database: "+word);
};
```

Створимо новий миттєвий план `ProcessTranslationRequestPlanG1`, який реагує на всі вхідні повідомлення з типом `query_ref` і створює підзадачі для всіх (правильно відформатованих) запитів. Припускаючи, що план витягнув дії (переклад або додати) і мову напрямків від повідомлень,

наступний код може бути використаний для створення, відправлення та очікування мети:

Java

```
Goal goal = new Goal (null, action + "" + dir,
Goal.TYPE_ACHIEVE, null, null);
goal.setParameter ("word", word);
goalfilter = dispatchSubgoalAndWait (goal);
```

Goalfilter повинна бути оголошена як змінна об'єкта плану, тому що вона може бути використана для перевірки вхідної події як результату відправлення мети. Процес плану перекладу повинен вивести результат його запиту. Інший блок коду в методі дії повинен звертатися до готової мети, виводячи результат. Цей блок коду повинен мати умову if (goalfilter.filter (event)). Цей фільтр був створений всередині і точно відповідає напрямку підцілі.

Створимо новий план EnglishFrenchTranslationPlanG1.

Створимо getEventFilter () метод так, щоб він реагував на спеціальну назву мети ("translate_english_french").

Java

```
PropertyFilter pf = new PropertyFilter ();
pf.addValue (Event.CLASS, GoalEvent.class);
pf.addValue ("type", GoalEvent.TYPE_PROCESS);
PropertyFilter gf = new PropertyFilter ();
gf.addValue ("name", "translate_english_french");
pf.addFilter ("goal", gf);
```

Реалізуйте метод дії так, щоб він витягав слово з мети (використовуйте goal.getParameter ("word")). Після виконання запиту на wordtable, встановити результат і результат стану цілі (using goal.setState (Goal.STATE_SUCCEEDED / FAILED) і goal.setResult (fword)).

Далі необхідно змінити план EnglishGermanTranslationPlanG1 так, щоб він реагував на ціль з ім'ям "translate_english_german". Крім того необхідно змінити метод дії так, щоб він отримував параметр від мети і оновлював результат цілі і результат стану.

Далі необхідно змінити ADF-файл, додати нове оголошення плану і зіставлення з ADF.

4.3 Завдання до роботи

4.3.1 Ознайомитися з літературою та основними теоретичними відомостями, необхідними для виконання роботи.

4.3.2 Ознайомитися з особливостями goal-орієнтованого програмування.

4.3.3 Узгодити з викладачем індивідуальне завдання на основі розширення агента з лабораторних робіт № 2-3. Реалізація завдання має ґрунтуватися на наступних особливостях: агент повинен мати певну мету, на яку буде направлена вся його діяльність, необхідно забезпечувати використання подій.

4.3.4 Визначити та описати мету агента і події у відповідності з його планами у вигляді тексту.

4.3.5 Із використанням бібліотеки JADDEX реалізувати агента, що буде розв'язувати поставлене завдання.

4.3.6 Виконати тестування та проаналізувати роботу створеного агента.

4.3.7 Оформити звіт з роботи.

4.3.8 Відповісти на контрольні запитання.

4.4 Зміст звіту

4.4.1 Мета роботи.

4.4.2 Опис переконань агентів, планів дій, цілей агентів.

4.4.3 Текст розробленого програмного забезпечення з коментарями.

4.4.4 Результати тестування агента.

4.4.5 Результати роботи програмного забезпечення.

4.4.6 Висновки, що відображають особисто отримані результати виконання роботи та їх критичний аналіз.

4.5 Контрольні запитання

4.5.1 У чому полягає goal-орієнтоване програмування?

4.5.2 У чому полягає зв'язок між goal-орієнтованим та агентно-орієнтованим програмуванням?

4.5.3 Які особливості goal-орієнтованого програмування в JADEX?

4.5.4 Якими властивостями повинен володіти адаптивний агент?

4.5.5 Чим відрізняються desires від goals?

4.5.6 Які види цілей існують та чим вони відрізняються?

4.5.7 Яким чином реалізуються різні типи цілей в Jadex?

4.5.8 Що таке мета-ціль?

4.5.9 Що розуміють під терміном «подія» і для чого вони застосовуються?

5 ЛАБОРАТОРНА РОБОТА № 5 ВЗАЄМОДІЯ З ЗОВНІШНІМИ ПРОЦЕСАМИ В JADEX

5.1 Мета роботи

5.1.1 Навчитися працювати із зовнішніми процесами з використанням платформи JADEX.

5.1.2 Вивчити принципи використання сокетів для виконання мережевої взаємодії.

5.2 Короткі теоретичні відомості

Часто виникає необхідність створення серверної частини, яка б очікувала певних запитів від клієнтської частини системи. Для подібної взаємодії в рамках мережі необхідно використовувати інтерфейс сокетів. Для забезпечення взаємодії через сокети необхідно працювати із зовнішніми процесами. Для цього потрібно оголосити:

Java

```
protected ServerSocket server;
```

У конструкторі необхідно додати:

Java

```
try {  
    this.server = new ServerSocket(port);  
}  
catch(IOException e) {  
    throw new RuntimeException(e.getMessage());  
}  
getLogger().info("Created: "+server);
```

Крім того, необхідно створити метод `close()` для вимкнення серверу:

Java

```
public void close() {
    try {
        getExternalAccess().getLogger().info("Closing: "+server);
        server.close();
    }
    catch(IOException e) {
        e.printStackTrace();
    }
}
```

У тілі класу слід додати частину, що буде очікувати підключення від клієнтської частини:

Java

```
new Thread(this).start();
getScope().addAgentListener(new IAgentListener() {
    public void agentTerminating(AgentEvent ae) {
        close();
    }
}, false);
```

У методі, що буде виконуватися в процесі запуску потоку, необхідно реалізувати обробник, наприклад:

Java

```
while (true)
{
    Socket client = server.accept();
```

Java

```

IGoal goal =
getExternalAccess().getGoalbase().createGoal("translate");
    goal.getParameter("client").setValue(client);

getExternalAccess().getGoalbase().dispatchTopLevelGoal(goal);
}

```

Для отримування мети та зчитування з потоку вводу від клієнтської частини можна зробити наступне:

Java

```

Socket client = (Socket)getParameter("client").getValue();
    BufferedReader in = new BufferedReader(new
    InputStreamReader(client.getInputStream()));
    String request = in.readLine();

```

Відправити назад відповідь клієнтові:

Java

```

    PrintStream out = new PrintStream(client.getOutputStream());
    out.print("HTTP/1.0 200 OK\r\n");
    out.print("Content-type: text/html\r\n"); out.println("\r\n");
    out.println("<html><head><title>TranslationG1
"+eword+"</title></head><body>");
    out.println("<p>Translated from English to German: "+eword+"
="+gword+".");
    out.println("</p></body></html>");
    client.close();

```

Введення типу мети:

ADF

```
<achievegoal name="translate">
  <parameter name="client" class="java.net.Socket"/>
</achievegoal>
```

Далі необхідно ввести новий план з налаштування сервера і запустити ініціалізацію плану:

ADF

```
<plan name="server">
  <parameter name="port" class="int">
    <value>9099</value>
  </parameter>
  <body class="ServerPlanG1"/>
</plan>
...
<configurations>
  <configuration name="default">
    <plans>
      <initialplan ref="server"/>
    </plans>
  </configuration>
</configurations>
```

Для того щоб план реагував на мету, необхідно змінити тригер плану та додати параметр для клієнта:

ADF

```
<plan name="egtrans">
  <parameter name="client" class="Socket">
    <goalmapping ref="translate.client"/>
  </parameter>
```

ADF

```

<body class="EnglishGermanTranslationPlanG1"/>
  <trigger>
    <goal ref="translate"/>
  </trigger>
</plan>

```

5.3 Завдання до роботи

5.3.1 Ознайомитися з літературою та основними теоретичними відомостями, необхідними для виконання роботи.

5.3.2 Ознайомитися з можливостями взаємодії з зовнішніми процесами в JADEX.

5.3.3 Узгодити з викладачем індивідуальне завдання на основі завдань лабораторних робіт № 2-4.

5.3.4 Завершити реалізацію програмної системи у відповідності з індивідуальним завданням та реалізувати мережеву взаємодію в межах даної системи. Мережева взаємодія має забезпечуватися за допомогою інтерфейсу сокетів. Мережева взаємодія має бути забезпечена також між користувачем та агентом. Перевірити представлений програмний код.

5.3.5 Виконати тестування розробленого програмного забезпечення.

5.3.6 Оформити звіт з роботи.

5.3.7 Відповісти на контрольні запитання.

5.4 Зміст звіту

5.4.1 Мета роботи.

5.4.2 Текст розробленого програмного забезпечення.

5.4.3 Результати тестування агента.

5.4.4 Результати роботи програмного забезпечення.

5.4.5 Висновки, що відображають особисто отримані результати виконання роботи та їх критичний аналіз.

5.5 Контрольні запитання

5.5.1 Що таке сокети?

5.5.2 За допомогою яких засобів забезпечується взаємодія через сокети?

5.5.3 Які особливості використання зовнішніх процесів?

5.5.4 Які особливості клієнт-серверної взаємодії в межах JADEX?

5.5.5 Які особливості серіалізації та десеріалізації?

5.5.6 Які класи використовуються в якості слухачів агентів та для чого вони необхідні?

5.5.7 Які методи дозволяють керувати слухачами агентів?

5.5.8 Для чого використовуються багатоагентні системи? Чи можна дані задачі розв'язати за допомогою одного агента?

6 ЛАБОРАТОРНА РОБОТА № 6 РОЗРОБЛЕННЯ МУЛЬТИАГЕНТНОЇ СИСТЕМИ

6.1 Мета роботи

6.1.1 Вивчити основні принципи агентно-орієнтованої мови програмування 2APL.

6.1.2 Навчитися розробляти мультиагентні системи.

6.2 Короткі теоретичні відомості

2APL – агентно-орієнтована мова програмування, що полегшує реалізацію мультиагентних систем.

На мультитоагентному рівні 2APL забезпечує програмні конструкції для визначення мультиагентних систем у вигляді набору окремих агентів та набору середовищ, в яких вони можуть виконувати свої дії.

На індивідуальному рівні 2APL забезпечує програмні конструкції для реалізації когнітивних агентів на основі BDI-архітектури. Забезпечуються програмні конструкції для реалізації переконань агента, цілей, планів, дій, подій та набір правил, через які агент може вирішити, які дії виконувати.

2APL є модульною мовою програмування, що дозволяє інкапсулювати когнітивні компоненти модулів. 2APL підтримує як реактивних, так і проактивних агентів.

Для розроблення мультиагентних систем потрібно завантажити платформу 2APL, наприклад, за даним посиланням http://apapl.sourceforge.net/?page_id=7.

Запустити платформу можна або використовуючи командний рядок, або використовуючи Eclipse.

Для запуску з командного рядка необхідно ввести наступну команду:

CMD

```
java -jar 2apl.jar
```

Після цього на екрані з'явиться вікно налаштування (рис. 6.1), з якого можна обрати режим подальшої роботи: індивідуальний агент, запуск над платформою Jade (для запуску мультиагентної програми).

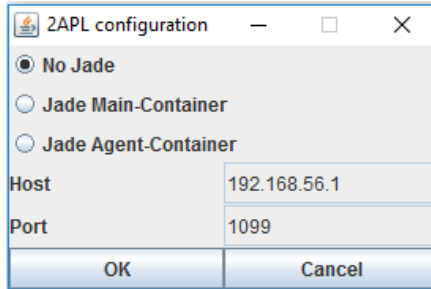


Рисунок 6.1 – Конфігурація 2APL

Приклад роботи з платформою 2APL представлено на рис. 6.2.

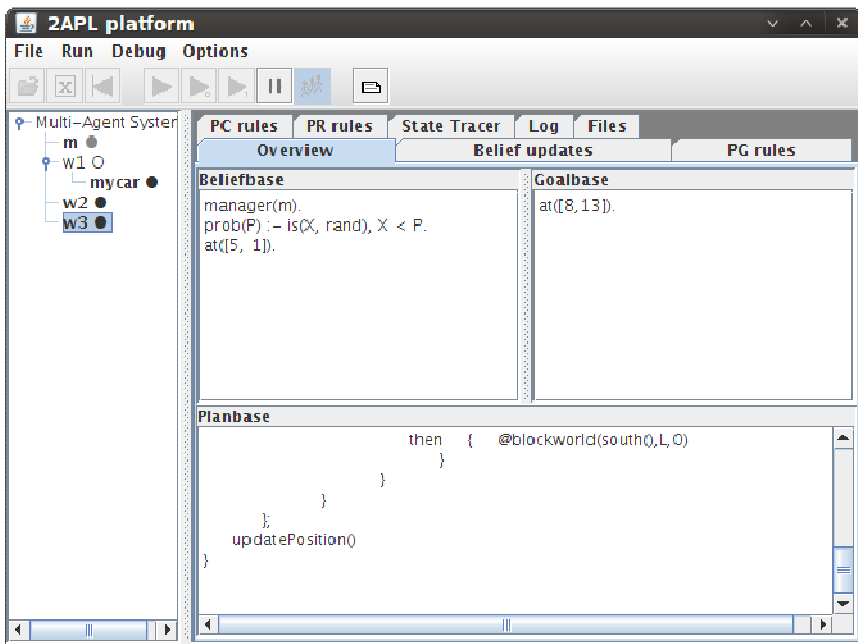


Рисунок 6.2 – Платформа 2APL

Для запуску 2APL з використанням Eclipse необхідно створити новий проект Java, після чого додати зовнішній JAR-файл (2apl.jar).

Далі додаток можна запустити, використовуючи клас APAPL.

Можна також інсталивати плагін для Eclipse, після чого можна буде створювати APAPL проекти.

Для розроблення мультиагентної системи для платформи 2APL необхідно створити три файли:

- файл специфікації мультиагентної системи (*.mas), що є XML-специфікацією;
- файл агента 2APL (*.2apl), що є текстовим файлом, який містить 2APL-код агента;
- файл середовища (*.jar).

Приклад mas-файлу:

MAS

```
<apaplmass>

  <environment name="blockworld" file="blockworld.jar">
    <parameter key="gridWidth" value="18" />
    <parameter key="gridHeight" value="18" />
  </environment>

  <agent name="m" file="manager.2apl" />
  <agent name="w1" file="worker.2apl" />
  <agent name="w2" file="worker.2apl" />
  <agent name="w3" file="worker.2apl" />

</apaplmass>
```

Приклад 2APL-файлу, в якому реалізовується один із агентів (manager), описаних в mas-файлі:

2APL

beliefs:

```
worker(w1).
worker(w2).
worker(w3).
```

goals:

```
haveBomb.
```

pgrules:

```
haveBomb <- bomb(POS) and not assigned(carrier(POS), _)
and
worker(W) and not assigned(_, W) |
{ send(W, request, play(carrier, POS));
+assigned(carrier(POS), W);
}
```

```
haveBomb <- not bomb(_) and worker(W) and
not assigned(_, W) |
{ send(W, request, play(explorer));
+assigned(explorer, W);
}
```

pcrules:

```
message(A, inform, _, _, bomb(POS)) <- true |
{ SetBomb(POS);
-assigned(explorer, A);
}
```

```
message(A, inform, _, _, done(POS)) <- true |
{ RemoveAssigned(carrier(POS), A);
-bomb(POS);
}
```

```
message(A, inform, _, _, failed(POS)) <- true |
{ -assigned(carrier(POS), A); }
```

Платформа 2APL містить набір інструментів для моніторингу виконання агентів, зокрема один з них представлений на рис. 6.3.

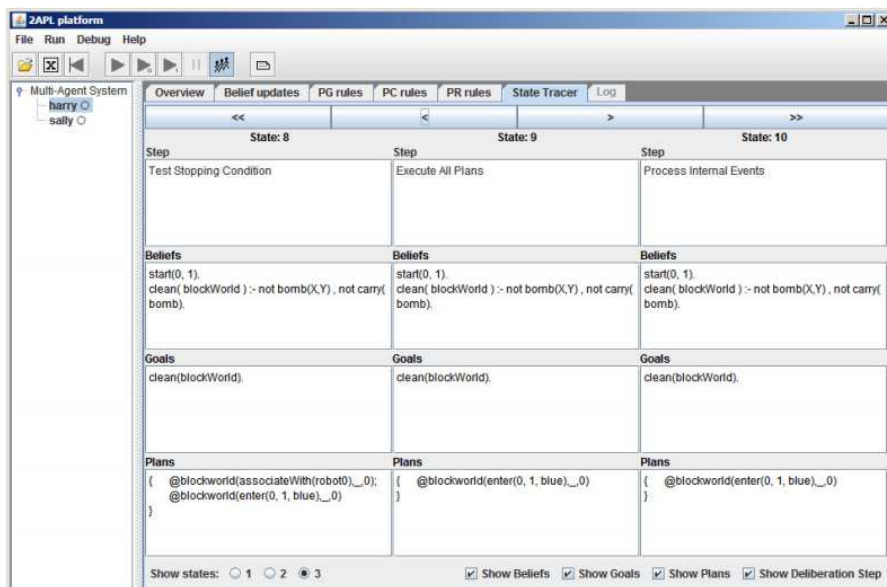


Рисунок 6.3 – Трасування агента під платформою 2APL

6.3 Завдання до роботи

6.3.1 Ознайомитися з основними теоретичними відомостями за темою роботи.

6.3.2 Узгодити з викладачем індивідуальне завдання.

6.3.3 Розробити мультиагентну систему у відповідності з індивідуальним завданням, використовуючи мову програмування 2APL.

6.3.4 Виконати тестування розробленої системи.

6.3.5 Оформити звіт.

6.3.6 Відповісти на контрольні запитання.

6.4 Зміст звіту

6.4.1 Мета роботи.

6.4.2 Текст розробленого програмного забезпечення.

6.4.3 Результати тестування.

6.4.4 Результати роботи програмного забезпечення.

6.4.5 Висновки, що відображають особисто отримані результати виконання роботи та їх критичний аналіз.

6.5 Контрольні запитання

6.5.1 Для чого необхідна мова програмування 2APL?

6.5.2 Яким чином можна запустити платформу 2APL та які режими роботи вона підтримує?

6.5.3 Яким чином визначити агентів, які беруть участь в мультиагентній взаємодії?

6.5.4 З яких файлів складається розроблення мультиагентної системи під платформою 2APL?

6.5.5 Які інструменти можуть бути використані для моніторингу виконання агентів під платформою 2APL?

6.5.6 Яким чином визначаються переконання та цілі агента в мові 2APL?

6.5.7 Яким чином можна змінювати переконання агента в процесі роботи?

6.5.8 Яким чином відбувається взаємодія між агентами?

6.5.9 Яким чином визначаються плани та яким чином визначити мови, за яких має виконуватись той чи інший план?

6.5.10 Що таке процедурні правила, яким чином вони визначаються та для чого необхідні?

7 ЛАБОРАТОРНА РОБОТА № 7 РОЗРОБЛЕННЯ WEB-АГЕНТА

7.1 Мета роботи

7.1.1 Вивчити основні принципи роботи програмних агентів.

7.1.2 Навчитися створювати web-агента, який взаємодіє зі службами в середовищі Internet.

7.2 Короткі теоретичні відомості

7.2.1 Основні поняття про програмні агенти

Агент – це апаратна або програмна сутність, здатна діяти в інтересах досягнення цілей, поставлених перед нею користувачем.

В межах мультиагентної парадигми програмні агенти представляють собою автономні компоненти, що діють від імені користувача.

Агентів також називають розумними агентами (Intelligent agent), тому що розумність є ключовим чинником при їхньому створенні. Хоча агенти можуть приймати різні форми, але вони всі реалізують якісь корисні функції замість виконання їх людиною. Агент може одержати завдання, яке необхідно виконати, і приймати потрібні рішення в процесі комунікації з іншими агентами (можливо, навіть із агентами, що представляють інтереси інших користувачів).

7.2.2 Властивості програмних агентів

Агенти можуть мати одну або декілька властивостей, наведених у таблиці 7.1.

Однією з основних властивостей, які асоціюються з програмними агентами, є автономність (Autonomy). Агент вважається автономним, якщо він здатний діяти без прямого керування людиною. Поняття автономності включає також наявність в агента декількох цілей, яких він може досягти. Наявність задач також сприяє тому, що агент одержує можливість планувати.

Таблиця 7.1 – Властивості програмних агентів

Властивість	Визначення
Автономність	Можливість діяти незалежно від користувача
Адаптивність	Здатність до навчання під час роботи
Комунікативність	Здатність до комунікації з користувачем або іншими агентами
Здатність до співробітництва	Працює з іншими агентами для досягнення мети
Персоніфікованість	Поводиться природно (проявляє емоції)
Мобільність	Можливість переміщатися по навколишньому середовищу

Агент є адаптивним (Adaptivity), якщо він може змінювати своє поведіння на підставі досвіду. Він повинен учитися й робити висновки залежно від стану навколишнього середовища й своїх знань.

Ще однією важливою характеристикою агентів є здатність до комунікації (Communicative). Агент повинен уміти спілкуватися з користувачем, щоб визначати свої завдання або ідентифікувати початкову інформацію. Крім того, агентові необхідно спілкуватися з навколишнім середовищем. Наприклад, пошуковий агент, що знаходить цікаві Web-сторінки, повинен уміти спілкуватися за допомогою протоколу HTTP, щоб підключатися до серверів і одержувати від них інформацію.

Агент здатний до співробітництва (Collaborative), якщо він може спілкуватися з іншими агентами для вирішення своїх завдань. Наприклад, агенти, які діють від імені користувачів на аукціонах, можуть брати участь в укладанні спільних угод, орієнтуючись на інформацію, отриману від користувачів. Подібні агенти звичайно входять у системи з більшою кількістю агентів, тому що для співробітництва потрібна безліч агентів.

Для деяких агентів найбільш істотною є здатність до персоналізації (Personality). Програми з їхнім застосуванням зазвичай використовуються в розважальних цілях. Тому, якщо персоналізація агента буде змінювати його залежно від переданої інформації, користувач зможе зрозуміти те, що агент намагається повідомити.

Деякі агенти мають мобільність (Mobility), або можливість переміщення по навколишньому середовищу. Ця властивість є дуже корисною при створенні певних програм. Властивість мобільності використовується у програмних агентах, що одержують інформацію з віддаленої бази даних. За звичайних умов хост приймає інформацію від бази даних, збирає потрібні відомості, а потім фільтрує їх за допомогою критеріїв, заданих користувачем. При одержанні мобільності агент може бути спрямований у віддалену базу даних, щоб автоматично відфільтрувати результати, а потім повернути тільки ті, які необхідні користувачеві. Це дозволяє як заощадити на пропускній здатності каналу, так і спростити архітектуру, особливо в тому випадку, якщо два хости можуть бути від'єднані один від одного.

Агент необов'язково повинен мати всі властивості. У більшості випадків використовуються всього дві.

7.2.3 Класифікація агентів

Незалежні агенти можна класифікувати наступним чином.

1. Агенти штучного життя.
2. Програмні агенти.
 - 2.1. Агенти для завдань.
 - 2.2. Агенти для проведення дозвілля.
 - 2.3. Віруси.
3. Інші агенти.

Агент, використання якого залежить від завдання, застосовується для розв'язання певної проблеми. Прикладом може служити пошуковий агент у мережі Internet. Крім пошуку за заданими критеріями, агент також стежить за тим, які зі знайдених посилань відкриє користувач. Це дозволяє створити шкалу цінностей пошуку, що сприяє кращому сортуванню результатів, а також прискорює одержання користувачем потрібного результату. Пошуковий агент може працювати у фоновому режимі, повторюючи пошук, щоб знайти й надати користувачеві нову інформацію.

Пошукові агенти володіють не тільки автономністю, але й адаптивністю, оскільки вони вивчають найцікавіші для користувача теми, ґрунтуючись на його реакції (тобто посиланнях, які відкриває користувач).

Іншим прикладом може служити агент, що використовується для роботи на аукціоні. Він здатний діяти від імені користувача, купуючи товари в мережі Internet за найнижчою ціною. Він може робити це в співробітництві з іншими агентами. Знайшовши агентів, які намагаються купити схожі товари, можна зробити спільну оптову покупку групи товарів за зниженою ціною.

Розважальні агенти корисні для взаємодії у віртуальному світі або подання персонажа в якості інтерфейсу користувача.

Вірус по-іншому можна назвати небезпечним мобільним програмним агентом. Хоча вірус не обов'язково повинен бути адаптивним, він є автономним і мобільним. Замість того щоб використовувати спеціальний протокол для переміщення мережею, вірус застосовує стандартні протоколи й найчастіше протокол передачі пошти.

Хоча зробити агент розумним досить складно, існує безліч методів, які можна використовувати для того, щоб додати агентові здатність до прийняття розумних рішень. До таких методів належать:

- нейронні мережі;
- системи, засновані на знаннях;
- дерева рішальних правил;
- еволюційні методи;
- нечітка логіка;
- алгоритми кластеризації.

7.2.4 Web-агент

Завдання Web-агенту полягає в тому, щоб взаємодіяти з певними службами в мережі Internet (наприклад, зі службами новин) і збирати інформацію на підставі критеріїв, заданих користувачем. Після збирання інформації Web-агент надає її користувачеві в тому порядку, який найбільш відповідає запиту (починаючи з даних, що мають найвищий рейтинг за умовами пошуку).

Web-агент використовує стандартні протоколи й передає користувачеві інформацію через звичайний браузер. Для спілкування з користувачем агент містить HTTP-сервер, для зв'язку із зовнішніми серверами є HTTP-клієнтом, а для одержання новин функціонує як клієнт протоколу NNTP (рис. 7.1).

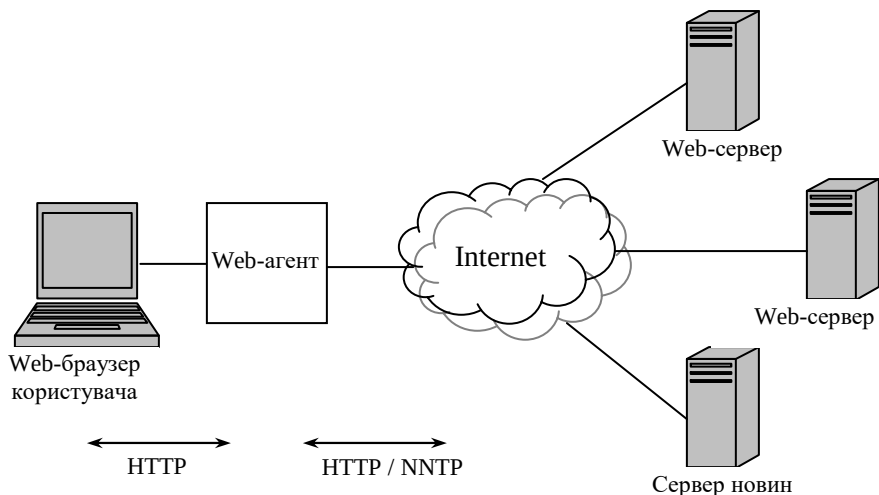


Рисунок 7.1 – Архітектура Web-агенту

Web-агент буде відігравати роль додатка, що здійснює взаємодію між користувачем і мережею Internet при читанні новин. Користувач змінює параметри агента за допомогою простого файлу конфігурації. У ньому користувач вказує web-сторінки для моніторингу (для видачі повідомлення у випадку їхньої зміни) і групи новин для перегляду й визначає специфічні критерії пошуку. Коли агент знаходить об'єкти, які відповідають заданому критерію, він зберігає їх і видає користувачеві у вигляді web-сторінки, посилання на якій відсортовані в потрібному порядку. Користувач може натиснути на посилання на певний web-сайт або рядок новин і перейти до потрібної статті.

Завдання агента полягає в тому, щоб спростити читання новин, відфільтрувати непотрібну інформацію й подати новини в порядку, за якого найцікавіші дані розташовані на першому місці.

Web-агент є дуже простим і не має здатності до навчання. Він володіє двома властивостями: автономністю (здатний працювати у фоновому режимі без прямого керування з боку користувача) і здатністю до комунікації з зовнішніми серверами при зборі інформації.

Сенсорами агента є стандартні протоколи, які дозволяють йому збирати потрібну інформацію за заданими критеріями пошуку. Для передачі результатів пошуку користувачеві використовується протокол NNTP. Потім користувач підключається до web-агента, як до будь-якого іншого web-сервера.

Для отримання інформації з середовища Internet web-агент виступає в якості NNTP-клієнта, який реалізує набір інтерфейсів для взаємодії з Internet-серверами, наприклад, із серверами новин. Ці інтерфейси дозволяють додатку з'єднуватись із сервером новин, задавати групи новин за певними критеріями, переходити до заголовку статті, зчитувати всю статтю, аналізувати повідомлення, пропускати поточну новину та переривати зв'язок із сервером новин.

Протокол NNTP є інтерактивним протоколом, який повністю базується на ASCII-командах. Нижче наведено приклад роботи із NNTP-сервером новин yellow.geeks.org за допомогою Telnet. Дані, що вводить користувач, виділені жирним текстом.

```
root@plato /root]# telnet yellow.geeks.org 119
S: 201 plato.mtjones.com DNEWS Version 5.5dl, S0, posting OK
C: list
S: 215 list.of newsgroups follows
S: control 2 3 y
S: control.cancel 2 3 y
S: my.group 10 3 y
S: new.group 6 3 y
S: .
C: group my.group
S: 211 8 3 10 my.group selected
C: article 3
S: 220 3 <3C36AF8E. 1BD3047E@mtjones.com> article retrieved
S: Message-ID: <3C36AF8E. 1BD3047E@mtjones.com>
S: Date: Sat, 05 Jan 2002 00:47:27 -0700
S: From: "M. Tim Jones" <mtj@mtjones.com>
S: X-Mailer: Mozilla 4.74 [en] (Win98; U)
S: X-Accept-Language: en
S: MIME-Version: 1.0
S: Newsgroups: my.group
S: Subject: this is my post
S: Content-Type: text/plain; charset=us-ascii
S: Content-Transfer-Encoding: 7bit
S: NNTP-Posting-Host: sartre.mtjones.com
S: X-Trace: plato.mtjones.com 1010328764 sartre.mtjones.com
(6 Jan 2002 07:52:44 -0700)
```

```

S: Lines: 6
S: Path: plato.mtjones.com
S: Xref: plato.mtjones.com my.group:3
S:
S:
S: Hello
S:
S: This is my post.
S:
S:
S: .
C: date
S: 111 20020112122419
C: quit
S: 205 closing connection - goodbye!

```

У наведеному прикладі показано, як з'єднання з NNTP-сервером створюється за допомогою Telnet-клієнта. Сервер новин відповідає вітанням, у якому вказується тип сервера й інша подібна інформація. Тепер через з'єднання можна давати команди. Існують два базові типи відповідей, які можна очікувати від сервера – з одного рядка й з декількох рядків. Щоб ідентифікувати кінець відповіді, протокол NNTP, як і протокол SMTP, застосовує символ “.” у рядку, (наприклад, відповідь на команди list і article).

7.3 Завдання до роботи

7.3.1 Ознайомитися з основними теоретичними відомостями за темою роботи.

7.3.2 Розробити програмного Web-агента, який взаємодіє з певними службами в Internet та збирає інформацію на основі критеріїв, що задані користувачем.

Програмний Web-агент має відповідати вимогам, наведеним в п. 7.3.2.1 – 7.3.2.5, і вимогам індивідуального завдання, узгодженого з викладачем, що визначає зокрема проблемну область, пошук новин з якої реалізується, критерії пошуку, та повинен бути написаним мовою програмування Java.

7.3.2.1 За допомогою протоколу HTTP Web-агент має перевіряти зміни заданих web-сайтів, на яких відображається потрібна інформація (наприклад, новини). При цьому зв'язок з сайтами організовується за допомогою інтерфейсу сокетів. Протокол

мережного рівня – IP, протокол транспортного рівня – TCP, протокол прикладного рівня – HTTP (порт 80). Перевірку на зміну сайтів з новинами рекомендовано виконувати шляхом перевірки розміру файлу, що пропонується завантажити.

7.3.2.2 За допомогою протоколу NNTP Web-агент має відбирати на вказаних серверах потрібну інформацію (наприклад, на сайті новин – статті), яка відповідає критеріям, що задані користувачем. Критеріями можуть бути: група новин та окремі слова, що мають зустрічатися в новині. Зв'язок з сервером новин має бути організований за допомогою інтерфейсу сокетів. Протокол мережного рівня – IP, протокол транспортного рівня – TCP, протокол прикладного рівня – NNTP (порт 119). При цьому врахувати наступні особливості серверів новин:

- після підключення до серверу має бути повернуто повідомлення, що вміщує код 201 у випадку вдалого підключення;

- для встановлення заданої групи новин необхідно відправити команду: `group назва_групи`. При цьому за досягнення успіху в результаті має бути повернуто повідомлення, що вміщує наступну послідовність: код 211, загальну кількість повідомлень даної групи, номер першого та останнього повідомлення;

- для завантаження заголовку повідомлення необхідно відправити команду: `head номер_повідомлення`. При цьому, якщо все було виконано успішно, то повертається код 221 та заголовок повідомлення;

- для завантаження усього повідомлення необхідно відправити серверу команду: `article номер_повідомлення`. Якщо повідомлення успішно передано, то повертається код 220 та саме повідомлення;

- для ідентифікації кінця відповіді сервера використовується рядок з символом “.”

7.3.2.3 Web-агент має надавати користувачу web-інтерфейс. При цьому мають бути передбачені такі можливості:

- можливість перегляду поточної конфігурації (тобто конфігурацію можна переглянути, прописавши в web-браузері посилання наступного типу: `http://192.168.115.2:8080/conf.html` – у випадку, коли web-агент працює на комп'ютері з ір-адресою 192.168.115.2 на порті 8080);

- можливість перегляду списку HTTP-сайтів та NNTP-серверів новин, на яких відбулися зміни (наприклад,

<http://192.168.115.2:8080/index.html>). Тобто має бути відображена інформація у вигляді двох таблиць: таблиця HTTP-сайтів, що змінилися, та таблиця повідомлень на серверах новин. При цьому для цих повідомлень має бути відображена група, до якої вони належать, а також задані слова, що були знайдені в цих повідомленнях. Послідовність відображення новин має відповідати ступеню їхньої відповідності критеріям користувача;

- можливість перегляду повідомлення з серверу новин у web-браузері користувача;

- надати можливість користувачу вказати HTTP-сервери, що було переглянуто, тобто які вже не треба виводити у таблиці до їх наступної зміни.

7.3.2.4 Робота зі збору інформації про стан серверів та робота з користувачем мають не заважати одна одній.

7.3.2.5 Конфігурація роботи web-агента має розміщуватися в окремому конфігураційному файлі, в якому можна вказати наступні параметри:

- http-сайти з новинами;
- сервери новин;
- групи новин та відповідні їм слова, що мають знаходитися в середині повідомлень або інші критерії.

7.3.3 Виконати тестування розробленого програмного агента.

7.3.4 Проаналізувати роботу створеного програмного агента. Виявити його переваги та недоліки.

7.3.5 Оформити звіт.

7.3.6 Відповісти на контрольні запитання.

7.4 Зміст звіту

7.4.1 Мета роботи.

7.4.2 Текст розробленого програмного забезпечення.

7.4.3 Результати тестування агента.

7.4.4 Результати роботи програмного забезпечення.

7.4.5 Висновки, що відображають особисто отримані результати виконання роботи та їх критичний аналіз.

7.5 Контрольні запитання

7.5.1 Що представляє собою програмний агент?

7.5.2 Поясніть та проаналізуйте властивості агентів: автономність, адаптивність, комунікативність, здатність до співробітництва, персоніфікованість, мобільність.

7.5.3 З якою метою застосовуються агенти, використання яких залежить від завдань?

7.5.4 Що таке розважальні агенти?

7.5.5 Проаналізуйте комп'ютерні віруси як програмні агенти.

7.5.6 Які методи використовуються для надання інтелектуальності програмним агентам?

7.5.7 Які протоколи використовує web-агент?

7.5.8 Виконайте аналіз архітектури web-агенту.

7.5.9 Якими властивостями володіє web-агент?

7.5.10 Проаналізуйте протокол, який використовується для передачі користувачу результатів пошуку за допомогою web-агенту.

7.5.11 Наведіть і поясніть схему потоків усередині web-агенту.

7.5.12 Проаналізуйте основні функції обраного інструментарію розробки web-агенту: внутрішня структура, параметри, основні змінні, їх призначення та використання.

ЛІТЕРАТУРА

1. Bullnheimer, B. Applying the ant system to the vehicle routing problem [Текст] / B. Bullnheimer, R. F. Hartl, C. Strauss // Meta-Heuristics : Advances and Trends in Local Search Paradigms for Optimization. – Boston : Kluwer, 1998. – Pp. 109–120.
2. Costa, D. Ants can colour graphs [Текст] / D. Costa, A. Hertz // Journal of the Operational Research Society. – 1997. – № 48. – Pp. 295–305.
3. Dorigo, M. Optimization, Learning and Natural Algorithms [Текст] / M. Dorigo. – Milano : Politecnico di Milano, 1992. – 140 p.
4. Dorigo, M. Ant colonies for the traveling salesman problem [Текст] / M. Dorigo, L. M. Gambardella // BioSystems. – 1997. – № 43. – Pp. 73–81.
5. Gambardella, L. M. HAS-SOP [Текст] : An hybrid ant system for the sequential ordering problem / L. M. Gambardella, M. Dorigo. – Lugano : CH, 1997. – Pp. 237–255.
6. Gambardella, L.M. Macs-vrptw [Текст] : A multiple ant colony system for vehicle routing problems with time windows / L.M. Gambardella, E. Taillard, G. Agazzi // New Methods in Optimisation ; In D. Corne, M. Dorigo, and F. Glover, editors. – McGraw-Hill, 1999. – Pp. 63–79.
7. Maniezzo, V. Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem [Текст] / V. Maniezzo. – Bologna : Universita di Bologna, 1998 – 102 p.
8. Michel, R. An ACO algorithm for the shortest common supersequence problem [Текст] / R. Michel, M. Middendorf // New Methods in Optimisation. – McGraw-Hill, 1999. – Pp. 525–537.
9. Subbotin, S. Modifications of Ant Colony Optimization Method for Feature Selection [Текст] / S. Subbotin, A. Oleynik // The experience of designing and application of CAD systems in Microelectronics : Proceedings of the IX International Conference CADSM-2007 (20–24 February 2007). – Lviv : Publishing house of Lviv Polytechnic, 2007. – Pp. 493–494.
10. Джонс, М. Т. Программирование искусственного интеллекта в приложениях [Текст] / М. Т. Джонс. – М. : ДМК Пресс, 2004. – 312 с.

11. Олейник, А. А. Мультиагентный метод многомерной оптимизации [Текст] / А. А. Олейник // Интеллектуальный анализ информации : восьмая международная конференция ИАИ-2008 : сборник трудов. – К. : НТУУ “КПІ”, 2008. – С. 340–350.
12. JADEX Active Components [Электронный ресурс] / Режим доступа : <http://www.activecomponents.org/bin/view/About/New+Home>
13. Нимейер, П. Программирование на Java [Текст] / Патрик Нимейер, Дэниэл Леук ; [пер. с англ. М.А. Райтмана]. – М. : Эксмо, 2014. – 1216 с.
14. Bellifemine, F. L. Developing Multi-Agent Systems with JADE [Текст] / Fabio Luigi Bellifemine, Giovanni Caire, Dominic Greenwood. – Wiley, 2007. – 300 p.
15. 2APL : A Practical Agent Programming Language [Электронный ресурс] / Режим доступа : <http://apapl.sourceforge.net/>
16. Tutorials & Guides | Jade Site [Электронный ресурс] / Режим доступа : <http://jade.tilab.com/documentation/tutorials-guides/>
17. Use WEKA in your Java code [Электронный ресурс] / Режим доступа : <https://weka.wikispaces.com/Use+WEKA+in+your+Java+code>