V. Meytus

KNOWLEDGE REPRESENTATION IN INTELLECTUAL SYSTEMS

The paper defines the intelligence in question. First as the ability to simulate a domain environment – subject area, and store information about it in terms of knowledge – experience cognitive of intellectual system and, secondly, as in this model to solve problems that connect to subject area. System behavior based on these decisions. Semantics on proposed classification of knowledge can be used in the construction of intelligence. These are the basic knowledge and knowledge – relations, searching, models, types, crossings. This classification knowledge required in view of their use as components of the model that created the intellectual engine for the environment.

Keywords: intelligence, intelligent system, knowledge, classification of knowledge, semantics knowledge.

Матеріал надійшов 20.06.2015

УДК 004.43.457:519.226

Глибовець А. М., Паращин О. О.

ПОБУДОВА БАЗИ ЗНАНЬ ІНТЕЛЕКТУАЛЬНОГО АГЕНТА НА ОСНОВІ МЕРЕЖІ БАЙЄСА

У роботі досліджено особливості використання мультиагентних систем і міркувань на основі правил (Rule-based Reasoning, RBR) для підтримки процесу прийняття рішень з представленням даних предметної області у вигляді мережі Байєса. Демонструється розгортання мультиагентної системи з метою обґрунтування використання програмного середовища такого типу для підтримки ефективного процесу прийняття рішень.

Ключові слова: мультиагентні системи, міркування на основі правил, мережа Байєса.

Вступ

Наявність невизначеності в адекватному представленні багатьох предметних областей сучасного світу мотивує пошук нових методів штучного інтелекту для вирішення проблеми складності обробки і підтримки ефективного процесу прийняття рішень. Природним убачається тут застосування байєсових мереж (БМ), які знаходять широке застосування при прогнозуванні, класифікації та керуванні процесами різної природи [1]. Серед різноманіття підходів зосередимося на використанні мультиагентних систем (Multi-Agent Systems, MAS) [2–4] і міркувань на основі правил (Rule-based Reasoning, RBR) [5].

Наразі немає у вільному доступі MAS, яка б змогла підтримати процес прийняття рішень. У статті опишемо розроблений прототип такої мультиагентної системи для подання даних

у вигляді БМ, міркування на основі правил (RBR) та можливе оновлення мережі на основі відповідей користувача. Прототип повинен надавати ефективні засоби для: моделювання інформації про предметну область; оцінки наслідків критичних процесів у прийнятті рішень; контролю процесів, що відбуваються в предметній області; моделювання і передбачення еволюції системи. Важливою компонентою тут було і розширення функціональності мультиагентної системи, щоб вона включала міркування на основі правил (RBR).

Результати проведеної роботи можна використовувати як експертну систему з можливістю навчання. Дані подаються у вигляді БМ, що дозволяє задати ймовірнісний розподіл.

Огляд використаних технологій і методів

Для проектування і розробки нашої мультиагентної системи ми проаналізували стан існуючих мультиагентних платформ на предмет можливості зберігання знання агента на основі моделі міркування RBR, зокрема, у вигляді байєсівської мережі. Було обрано платформу JADE [6].

JADE – це проміжне програмне забезпечення, спрямоване на підтримку розробки, яка базується на одноранговій мережі інтелектуальних агентів. Воно дає змогу розробити систему агентів, які можуть активно працювати відповідно до поведінки, спілкуватись і вести переговори з іншими агентами, незалежно від їхньої ролі і позиції, координувати розподілене вирішення складних проблем.

Інтелектуальність системи, за Джеффрі Вулдріджом, визначатиметься інтелектуальністю агента: *агент являє собою комп'ютерну систему, що перебуває в деякому середовищі і здатна до автономної дії в цьому середовищі для виконання свого проектного завдання. Інтелектуальний агент додатково повинен бути автономний, реактивний, ініціативний та соціальний.*

Ключовим завданням в агентній архітектурі є балансування реактивності та ініціативності. З одного боку, агент повинен бути реактивним, оскільки на його плани і дії впливає навколишнє середовище. З другого боку, плани і дії агента мають визначатися його цілями. Завдання полягало у збалансуванні двох, часто суперечливих, джерел впливу: якщо агент занадто реактивний, то він постійно коригуватиме свої плани і не досягне мети. Однак якщо агент недостатньо реактивний, то він втрачатиме час, намагаючись дотримуватись планів, які більше не потрібні або не доречні.

Агенти зазвичай використовуються там, де середовище домену є складним завданням; більш конкретно, типове середовище агентів є динамічним, непередбачуваним і ненадійним.

Як відомо, основою для більшості моделей прийняття управлінських рішень є модель вирішення завдань Герберта Саймона. Ця модель зображує процес вирішення проблеми у вигляді потоків, які можуть оброблятись як лінійно, так і ітеративно, що дає змогу повернутись до попередніх кроків для додаткового уточнення.

Річард Сойда [5] визначає інтелектуальну систему підтримки прийняття рішень як систему, яка використовує комбінацію моделей, аналітичних методів, пошуку інформації для допомоги в розробці та оцінці альтернативи. Системи підтримки прийняття рішень повинні сприяти зменшенню невизначеності, з якою стикаються менеджери, коли їм потрібно прийняти рішення щодо майбутніх варіантів. Рішення треба приймати, коли спостерігається або прогнозується відхилення від очікуваного, бажаного стану системи. Це передбачає розуміння проблеми, що, своєю чергою, ґрунтується на інформації, досвіді, знаннях про процес.

Системи бізнес-правил фокусуються на поданні знань, наприклад, щоб подати логіку першого порядку в короткій, однозначній, зрозумілій формі. Така система традиційно використовує механізм виведення. Вона зіставляє факти і дані з правилами, щоб мати змогу отримати висновки, які спричиняють певні дії.

Зіставлення з шаблоном – це процес зіставлення нових або існуючих фактів з правилами. Системи логічного висновку використовують різні алгоритми зіставлення, а саме: Rete, Treat, Leaps. Ми обрали для застосування алгоритм Rete.

Останній може здійснювати зіставлення з шаблоном, який включає декілька тисяч робочих елементів у пам'яті, без будь-яких значних потреб ресурсів чи неприйнятних втрат продуктивності. Цей алгоритм використовує той факт, що вміст робочої пам'яті мало змінюється після кожного застосування правила, робить лише незначні зміни в порівнянні з попереднім проходом. Зокрема, алгоритм Rete з'ясовує, які правила не спрацювали в попередньому циклі, які правила з попереднього циклу не спрацюють у наступному, які правила з попереднього циклу, що не спрацювали, найімовірніше спрацюють у наступному циклі. Використовуючи цей метод, алгоритм уникає виконання циклу розпізнавання шаблонів щоразу з нуля.

Проектування та розробка системи

Розроблена система є програмним інструментом, який дозволяє в процесі прийняття рішень простежити причинно-наслідковий зв'язок у різних сценаріях. Для тестування системи було обрано проблему знаходження пошкодження мобільного телефону. Кінцевий користувач може знайти причину пошкодження, відповівши на серію запитань. На основі відповідей користувача і логічних міркувань система може вказати причину пошкодження.

Мультиагентна система

У системи є два джерела інформації: дані від користувача та представлення певної предметної області у вигляді БМ. Основну архітектуру системи зображено на рис. 1.

Представлення бази знань відбувається у два етапи: створення об'єктів предметної області (фактів) у вигляді POJO (Plain Old Java Objects), відповідно до специфікації JavaBean, і створення Drools Rule Language (DRL) файлу з правилами. Маємо створити спеціального агента DroolsAgent, який буде посередником між мультиагентною системою і системою бізнес-правил Drools, та визначити опис послуг, включаючи послугу міркування, які може надавати DroolsAgent. Описуючи створення поведінки агента DroolsAgent, маємо пам'ятати, що він може отримувати компоненти системи Drools, завантажувати робочу пам'ять з фактами користувача, отриманими від агента UserAgent. Він також може додавати нові факти до робочої пам'яті, дочекатись відповіді, щоб відправити результат користувачеві. Тобто DroolsAgent містить у собі екземпляр системи Drools і оновлює дані в БМ.

У системі передбачається робота ще двох груп агентів: UserAgent, PersistenceAgent.

UserAgent – група агентів, які співпрацюють з користувачем. Для кожного користувача, який входить у систему, створюється пара агентів – UserAgent та DroolsAgent, які завжди працюють у парі. Агент UserAgent отримує дані від користувача та запитує про нові факти. Після отримання всіх фактів та обрахування логічного факту користувачеві повертається результат.

PersistenceAgent – відповідає за збереження предметних областей у вигляді БМ у базі даних MongoDB. Це єдиний агент, який існує в контейнері JADE в єдиному екземплярі, оскільки пара агентів UserAgent, DroolsAgent створюється для кожного окремого користувача. Агент PersistenceAgent запускається одразу після старту контейнера JADE.

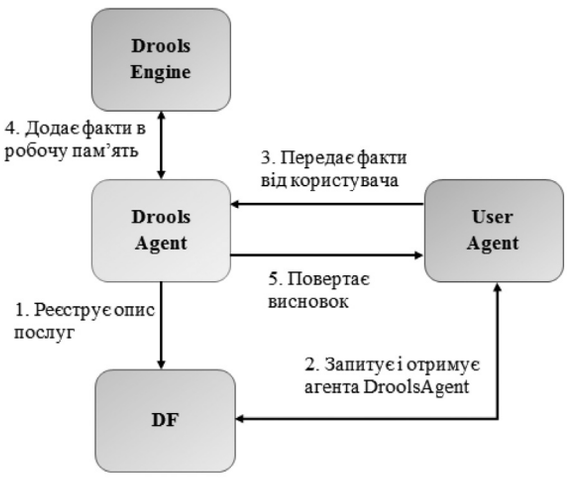

Рис. 1. Архітектура системи

Інтеграція системи Drools з платформою JADE

Ми використали розширення мультиагентної платформи JADE для забезпечення механізму виведення міркувань на основі правил. БМ, реалізована в системі Drools, була інтегрована в агента DroolsAgent, який здатний виконувати певні міркування.

Ідея полягала в тому, щоб використати платформу JADE, з її можливістю конфігурувати поведінку агентів, а також спілкування агентів для реалізації процесу міркування. Окрім цього, потрібно було забезпечити механізм логічного висновку шляхом реалізації агентів, які реалізують свою логіку на основі правил.

Представлення знань

Знання представляються у вигляді правил і фактів. Правила задаються в спеціальних DRL файлах, факти – JavaBean.

Факти – це об'єкти предметної області, написані на Java; вони представлені як POJO і реалізовані відповідно до специфікації JavaBeans. Факти створюються за допомогою конструктора класу. Клас повинен мати методи-селектори і методи-модифікатори для кожного поля, щоб система Drools мала можливість отримати значення та маніпулювати ним.

Ми заносимо правила в робочу пам'ять Drools; потім система правил Drools використовує ці факти для оновлення стану БМ.

JavaBeans – це Plain Old Java Objects, які можна серіалізувати та які мають конструктор без параметрів, надають доступ до властивостей за допомогою методів-селекторів і методів-модифікаторів. Мета JavaBean – інкапсуляція багатьох об'єктів в одному об'єкті (bean), отже, ці об'єкти можуть бути передані в одному об'єкті bean, а не як кілька окремих об'єктів.

Клас JavaBean має дотримуватись певних правил щодо найменування методів, конструктора та поведінки. Ці правила дозволяють існування інструментів, які можуть повторно використовувати, замінювати, з'єднувати JavaBean.

Правила представлені у DRL файлах. DRL файл – простий текстовий файл, який може містити правила, запити і функції. Правила написані з використанням логіки першого порядку.

Правило складається з двох частин: умова та наслідок. У Drools частина, в якій описано умову, називається лівостороння частина (Left Hand Side, LHS), частина, в якій описано наслідки, – правостороння частина (Right Hand Side, RHS). Ці дві частини формують правило у вигляді «якщо-тоді», яке в Drools представлене у вигляді «when-then».

Правило повинне мати унікальне ім'я в межах свого пакету. Атрибути є необов'язковими; вони впливають на поведінку правила. Атрибут «діалект» визначає мову, яка використовуватиметься для виразів у правилах, це може бути Java або Mvel.

Лівостороння частина правила може містити нуль або більше умов. Якщо в лівосторонній частині немає умов, така умова розглядатиметься як істина. Правостороння частина правила може містити список дій, які будуть виконані (наприклад, вставка, оновлення, видалення фактів).

Умова правила має на меті знаходження одного або декількох об'єктів фактів. Якщо факти, які містяться в робочій пам'яті Drools, задовольняють усі умови правила, то правило буде активовано і всі наслідки цього правила буде виконано. Наслідки, визначені в правилах, можуть оновити факти, що містяться в JavaBeans, виконати метод Java тощо.

DroolsAgent

Агент DroolsAgent виступає посередником між іншими агентами та системою Drools; архітектуру агента зображено на рис. 2. Агент DroolsAgent створює сервіс «Drools Service» з описом послуг, які він може надавати, та публікує його в DF (Directory Facilitator) JADE, який забезпечує послугу «жовтих сторінок», де агенти можуть публікувати послуги.

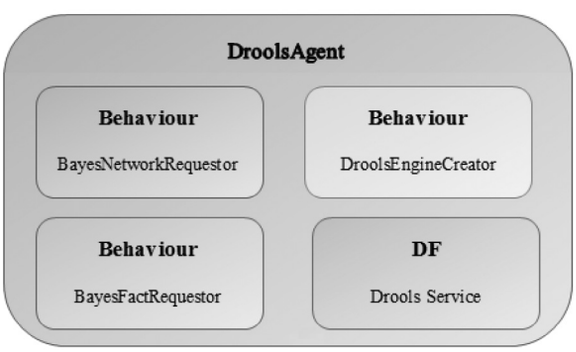

Рис. 2. Архітектура агента DroolsAgent

Протягом свого життєвого циклу агент DroolsAgent має кілька поведінок:

– BayesNetworkRequestor – одразу після ініціалізації агент UserAgent шукає свого агента DroolsAgent, щоб запитати, які можливі предметні області зберігаються в сховищі у вигляді БМ;

– DroolsEngineCreator – після того, як агент UserAgent отримав від користувача відповідь, яка предметна область його цікавить, агент DroolsAgent створює та ініціалізує DroolsEngine;

– BayesFactRequestor – на основі даних, які зберігаються в системі Drools (агента DroolsAgent), запитує в користувача нові факти, на основі яких оновлюватиметься інформація в байєсівській мережі.

Коли всі можливі факти внесено в робочу пам'ять системи Drools і стан БМ оновлено, результати (висновку) надсилаються користувачеві через агента UserAgent як посередника.

Інтеграція Spring

Для інтеграції в єдину систему контейнера JADE, системи Drools, БМ та графічного інтерфейсу ми використали фреймворк Spring.

З багатьох модулів фреймворку було використано лише кілька: контейнер інверсії управління, доступ до даних, модель-вигляд-управління, аутентифікація і авторизація.

Контейнер інверсії управління – це конфігурація компонентів додатків і управління життєвим циклом об'єктів Java, яка здійснюється головним чином через інверсію управління. Доступ до даних забезпечує роботу з об'єктно-реляційними відображеннями та інструментами з NoSQL баз даних за допомогою підпроекту Spring Data. Model-View-Controller представляє програмний каркас на основі HTTP-сервлета, що забезпечує створення веб-додатків і веб-служб RESTful за допомогою підпроекту Spring MVC. Аутентифікація і авторизація являють собою налаштовувані процеси безпеки, які підтримують цілий ряд стандартів, протоколів, інструментів і практик за допомогою підпроекту Spring Security.

Під час інтеграції каркасу Spring та JADE було використано бібліотеку Jade4Spring.

Для надання можливості керування контейнером JADE потрібно додати такий бін у конфігураційний файл контексту Spring:

```
<!--JADE Container bean -->
<bean id="jadeBean"
    class="net.sf.jade4spring.JadeBean"
    init-method="startContainer" lazy-init=
    ="false"
    destroy-method="stopContainer">
</bean>
```

Є кілька властивостей, на які потрібно звернути увагу. Властивість utilityAgents може приймати значення true або false, яке вказує, чи повинен контейнер запустити додаткові агенти, які спостерігають за станом контейнера, до якого вони під'єднані. Один з таких агентів – J4SInfiltratorAgent – відстежує всі події в платформі та стежить за активними агентами. JadeBean використовує цього агента для отримання інформації про платформу.

Додаткові агенти запускаються за замовчуванням, тому, щоб відключити їх, потрібно додати такий елемент до конфігурації JadeBean:

```
<property name="utilityAgents">
    <value>false</value>
</property>
```

Агенти визначаються в конфігурації контексту Spring таким чином:

```
<bean id="persistenceAgent"
    class="com.alex.diploma.bayesnetwork.
    PersistenceAgent"
    scope="singleton">
    <property name="repository" ref="
    bifRepository"/>
</bean>

<bean id="droolsAgent"
    class="com.alex.diploma.bayesnetwork.
    drools4jade.DroolsAgent"
    scope="prototype">
    <property name="repository" ref="
    bifRepository"/>
</bean>

<bean id="userAgent"
    class="com.alex.diploma.bayesnetwork.
    UserAgent"
    scope="prototype">
</bean>
```

Зауважимо, що агенти UserAgent і DroolsAgent мають значення scope="prototype". Це означає створення нового об'єкта для кожного біна, оскільки UserAgent і DroolsAgent створюються для кожного користувача. Агент PersistenceAgent має значення scope="singleton", оскільки цей агент існуватиме в системі в єдиному екземплярі.

Також Spring Data дозволяє легко сконфігурувати доступ до бази даних MongoDB. У файлі конфігурації це виглядає таким чином:

```
<mongo:mongo host="127.0.0.1"
port="27017"/>
<mongo:db-factory dbname="bayes"/>
<mongo:repositories base-package="com.alex.
diploma.bayesnetwork.repo"/>
<bean id="mongoTemplate"
    class="org.springframework.data.mongodb.
    core.MongoTemplate">
    <constructor-arg name="mongoDbFactory"
    ref="mongoDbFactory"/>
</bean>
```

MongoDB

MongoDB – документо-орієнтована система керування базами даних, яка не потребує опису схеми таблиць. Вона підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів, підтримує реєстрацію операцій зі зміни і додавання даних.

Усі ці переваги стали аргументом на користь нашого вибору MongoDB як сховища БМ. Оскільки в систему можна завантажувати нові БМ у XMLBIF форматі, мережу можна легко зберегти в базі, попередньо конвертувавши: XMLBIF -> JavaBean -> JSON.

XMLBIF (The Bayesian Interchange Format) – простий формат, який дозволяє зберігати різні БМ. Метою цього формату є представлення орієнтованого ациклічного графа, який може бути пов'язаний з умовною ймовірністю дискретних змінних.

Спілкування та взаємодія між агентами

У JADE зв'язок між агентами здійснюється за допомогою повідомлень. Зазвичай ці повідомлення між агентами відбуваються в рамках їхнього спілкування. Механізм обробки спілкування заснований на структурі повідомлень FIPA, де для управління розмовою використовуються три параметри: conversation-id (унікальне значення, яке групує кілька повідомлень, що відносяться до однієї розмови), in-reply-to (ідентифікує повідомлення як відповідь до попереднього повідомлення з відповідним значенням параметра reply-with), reply-with (ідентифікатор, який очікується у відповідь на це повідомлення).

FIPA визначила 22 комунікативних акти, кожен з чітко визначеною семантикою, які дозволяють задовольнити 95 % усіх можливих ситуацій. Для спілкування між агентами в системі використовуються такі дії:

– CFP – дія CFP (call for proposal, заклик до пропозиції) використовується для ініціювання переговорів між агентами. По суті, агент говорить: «Ось дія, яка повинна бути виконана, і ось умови, за яких ця дія має виконуватись, – чекаю на пропозиції».

– PROPOSE – ця дія дозволяє агенту зробити пропозицію іншому агенту у відповідь на повідомлення CFP, яке було надіслане раніше.

– ACCEPT-PROPOSAL – дія accept-proposal дозволяє агенту повідомити, що він приймає пропозицію, яку надіслав інший агент.

– REQUEST – одна з фундаментальних дій, дозволяє агенту попросити іншого агента виконати певну дію.

– PROPAGATE – вміст propagate повідомлення складається з двох частин: текст повідомлення та вираз, який позначає набір агентів. У нашому випадку отримувачем такого повідомлення буде лише один агент DroolsAgent.

Комунікативний акт між агентами UserAgent та DroolsAgent представлено на рис. 3.

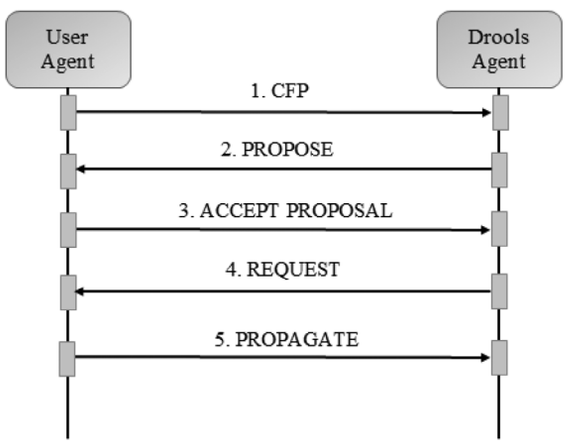

Рис. 3. Комунікативний акт між UserAgent та DroolsAgent

Моделювання предметної області

Для моделювання предметної області було обрано мобільні телефони. У знаходженні причини пошкодження телефону може допомогти розроблена система. Причини втрати сигналу можна подати у вигляді байєсівської мережі з урахуванням імовірності подій (рис. 4).

Як можна побачити, можливими причинами втрати сигналу є: проблеми з SIM-картою; проблема з тримачем SIM-карти; SIM-карта просто змістилась, тому немає сигналу. SIM-карта не працює, якщо телефон заблокований для конкретного оператора, закінчився термін дії, погані контакти.

Телефон упав або отримав інше механічне пошкодження, в результаті цього був пошкоджений тримач SIM-карти. Як наслідок – він не працює і зник сигнал зв'язку. У телефоні в результаті механічного втручання пошкоджено материнську плату. Як наслідок – з перебоями працюють усі системи телефону, зокрема тримач SIM-карти. У телефон потрапила волога і пошкодила материнську плату. Як наслідок – не працює тримач SIM-карти.

Телефон упав або отримав інше механічне пошкодження, в результаті чого SIM-карта змістилась і контакти SIM-карти перебувають у неправильному положенні, тому зник сигнал зв'язку.

На основі відповідей користувачів у БМ вносяться нові факти, на основі яких будується логічний висновок, що стало причиною пошкодження мобільного телефону.

Додавання нової предметної області

Система передбачає додавання нових предметних областей. Деякий привілейований користувач може завантажити нову предметну область у вигляді БМ, яку всі інші користувачі зможуть використовувати.

Формат XMLBIF забезпечує можливість зберігання та маніпулювання БМ. Замість того, щоб

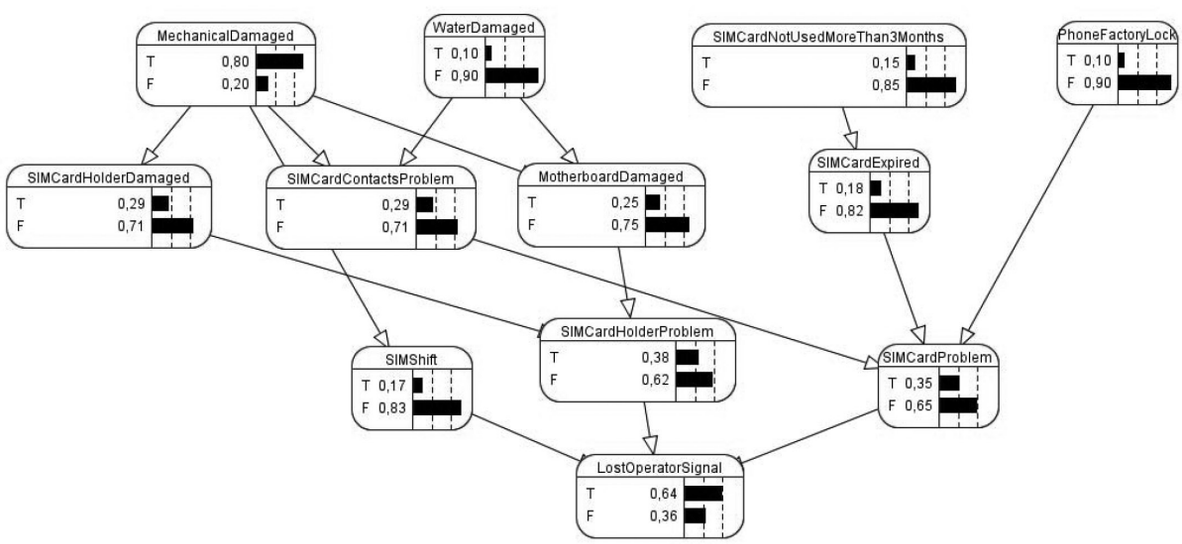

Рис. 4. Байєсівська мережа причин втрати сигналу

зосередитись на читанні опису байєсівської мережі, формат XMLBIF підкреслює простоту поширення через глобальні мережі.

Для побудови БМ у XMLBIF форматі використовувалась програма JavaBayes.

JavaBayes – набір інструментів, призначений для створення і маніпулювання БМ. Система складається з графічного редактора, механізму виведення і набору синтаксичних аналізаторів. Графічний редактор з дружнім інтерфейсом дозволяє створювати і редагувати БМ. Парсери дозволяють імпортувати БМ у різних форматах. Механізм виведення відповідає за зміну структури даних, які представляють БМ. Механізм виведення може обрахувати граничну ймовірність будь-якої змінної в БМ, очікувані значення для одновимірних функцій (наприклад, очікуване значення для змінної), конфігурації з максимальною апостеріорною ймовірністю.

JavaBayes може представляти БМ у XMLBIF форматі.

Висновки

У статті описано розроблену мультиагентну систему, здатну підтримувати процес прийняття рішень та міркування на основі правил. Система здатна мобільно налаштовуватись на підтримку процесу прийняття рішень в обраній предметній області реального світу.

У цілому система бере інформацію про характеристики і події предметної області з відповідей користувача. Як тільки система отримала достатньо знань, вона в змозі надати інформацію і пояснити результат.

Для того щоб досягнути поставлених цілей, мультиагентну платформу JADE було розширено шляхом інтеграції системи правил Drools, яка базується на алгоритмі Rete, в агента DroolsAgent, який здатний виконувати прийняття рішень та робити логічний висновок. Тестування прототипу розробленої системи підтвердило ефективність прийнятих рішень.

Список літератури

1. Бідюк П. І. Методика побудови динамічних мереж Байєса [Електронний ресурс] / П. І. Бідюк, О. О. Павлюк. – Режим доступу: old.bulletin.kpi.ua/files/2010-2-9.pdf. – Назва з екрана.
2. Padgham L. Developing Intelligent Agent Systems [Electronic resource] / Lin Padgham, Michael Winikoff. – RMIT University, Melbourne, Australia, 2004. – 221 p. – Mode of access: http://www.agilemethod.csie.ncu.edu.tw/download/agent/AOSEPrometheus.pdf. – Title from the screen.
3. Padgham L. Developing Intelligent Agent Systems: A Practical Guide / Lin Padgham, Michael Winikoff ; RMIT University. – Melbourne, Australia : John Wiley and Sons, 2004. – 241 p.
4. Salard T. R. A Multi-Agent System framework to support the decision-making in complex real-world domains [Electronic resource] / T. R. Salard. – Thania Rendón Sallard, 2009. – 56 p. – Mode of access: http://upcommons.upc.edu/bitstream/handle/2099.1/7723/Master%20Thesis%20-%20Thania%20Rendon%20Sallard.pdf?sequence=1. – Title from the screen.
5. Sojda R. S. Artificial intelligence based decision support for trumpeter swan management : PhD Dissertation [Electronic resource] / R. S. Sojda ; Colorado State University. – Fort Collins, Colorado, 2002. – 183 p. – Mode of access: http://www.nrmsc.usgs.gov/files/norock/products/Sojda_Dis.pdf. – Title from the screen.
6. JADE Architecture Overview. Tutorial [Electronic resource]. – Mode of access: http://jade.tilab.com/documentation/tutorials-guides/jade-administration-tutorial/architecture-overview. – Title from the screen.

A. Glybovets, O. Paraschyn

BUILDING KNOWLEDGE BASE OF INTELLIGENT AGENT BASED ON BAYESIAN NETWORKS

The paper investigates especially the use of multi-agent systems and considerations on the basis of the rules (Rule-based Reasoning, RBR) to support decision-making with the presentation of subject data as Bayesian networks. Demonstrated the deployment of multi-agent system in order to justify the use of the software environment of this type to support efficient process decision-making.

Keywords: multi-agent systems, Rule-based Reasoning (RBR), Bayesian networks.

Матеріал надійшов 15.10.2015