STUDIA INFORMATICA 2016
Volume 37 Number 1 (123)

Roman SIMIŃSKI
Uniwersytet Śląski, Instytut Informatyki

BIBLIOTEKA KBEXPERTLIB DLA JĘZYKA JAVA – WŁAŚCIWOŚCI FUNKCJONALNE I BADANIA WYDAJNOŚCIOWE[1]

Streszczenie. Biblioteka *KBExpertLib* jest nowym, w pełni autorskim pakietem, oprogramowaniem, oferującym większość typowych funkcji jądra szkieletowego systemu ekspertowego Biblioteka pozwala na budowanie systemów ekspertowych z wykorzystaniem języka Java. Celem niniejszej pracy jest prezentacja właściwości funkcjonalnych biblioteki *KBExpertLib* oraz wyników badań wydajnościowych najistotniejszy operacji przez nią oferowanych. Eksperymenty są skoncentrowane na badaniu wydajności czasowej podstawowych operacji biblioteki oraz ocenie jej wymagań pamięciowych. W trakcie badań wykorzystywane były rzeczywiste bazy wiedzy, w tym licząca ponad 20 000 reguł.

Słowa kluczowe: system ekspertowy, baza wiedzy, wnioskowanie

THE KBEXPERTLIB SOFTWARE LIBRARY FOR JAVA – FUNCTIONALITY PROPERTIES AND PERFORMANCE STUDY

Summary. The *KBExpertLib* is the originally designed software package which provides most of the expert system shell's common functions. This library allows to build expert systems using Java programming language. The presentation of the functional properties of the *KBExpertLib* is the goal of this paper as well as the performance study of basic activities provided by the library. The experimental research are focused on the time efficiency of the basic *KBExpertLib* operations. The estimation of the memory occupation of the library data structures are also presented. The experiment was performed on the real-world knowledge bases, counting more than 20 000 rules.

Keywords: expert system, knowledge base, inference

[1]Niniejsza praca jest współfinansowana ze środków Narodowego Centrum Nauki (NCN: 2011/03/D/ST6/03027, „Eksploracja regułowych baz wiedzy").

1. Wprowadzenie

Systemy ekspertowe są znane od wielu lat, a liczne, udokumentowane zastosowania udowodniły ich praktyczną użyteczność [1]. Mimo istnienia różnych metod reprezentacji wiedzy, najpopularniejszą i ciągle wykorzystywaną metodą reprezentacji okazała się reprezentacja regułowa. Swoisty renesans zainteresowania systemami regułowymi przyniósł rozwój eksploracji danych [2, 3], możliwość automatycznego [4] pozyskiwania reguł znaczenie ułatwiła budowanie systemów wykorzystujących regułowe bazy wiedzy. Projekt KBExplorator [7] ma dostarczyć podstaw teoretycznych, jak i praktycznych metod organizacji regułowych baz wiedzy, metod ich analizy i weryfikacji, a także zoptymalizowanych oraz nowych metod wnioskowania. Podstawy teoretyczne projektu *KBExplorator* zostały opisane w pracach [8-10], a wybrane aspekty implementacyjne w [11-13].

Praktycznym efektem zrealizowanych prac jest aplikacja internetowa *KBExplorator* oraz biblioteka programowa *KBExpertLib*. *KBExplorator* jest aplikacją dedykowaną dla inżynierów wiedzy, pozwala zarejestrowanym użytkownikom na tworzenie i edycję baz wiedzy. Biblioteka *KBExpertLib*, dedykowana dla języka Java, wykorzystuje bazy wiedzy systemu *KBExplorator*, oferuje ona klasy reprezentujące wszystkie esencjonalne elementy baz wiedzy oraz klasy pozwalające na realizację różnych metod wnioskowania – zarówno w wersjach klasycznych, znanych z literatury, jak i wersjach autorskich [8, 10]. Dzięki użyciu biblioteki *KBExpertLib* można w łatwy sposób rozszerzyć funkcje aplikacji tworzonych z zastosowaniem języka Java o elementy wykorzystujące regułowe bazy wiedzy i wnioskowanie.

KBExplorator oraz biblioteka *KBExpertLib* znajdują się w fazie poprzedzającej ich udostępnienie, które jest przewidywane we wrześniu 2016 roku. Biblioteka *KBExpertLib* znajduje się w fazie weryfikacji jej praktycznej użyteczności, skuteczności oraz oceny efektywności dla rzeczywistych baz wiedzy w różnych zastosowaniach. Badania obejmują eksperymenty realizowane w obrębie (i) klasycznych aplikacji klasy desktop, uruchamianych na komputerach PC (i ten aspekt właśnie jest rozważany w niniejszej pracy), (ii) w aplikacjach dedykowanych dla urządzeń mobilnych pracujących pod kontrolą systemu Android, (iii) w warstwie serwerowej aplikacji internetowych realizowanych z wykorzystaniem J2EE, udostępniającej usługi w architekturze REST. Docelowo przewiduje się również przeprowadzenie eksperymentów dla innych platform mobilnych niż Android oraz dla takich platform, jak Raspberry Pi oraz Arduino, wraz z analizą możliwości optymalizacji dla środowisk o ograniczonych zasobach.

W niniejszej pracy postawiono dwa, powiązane ze sobą cele, a mianowicie pierwszym jest przedstawienie informacji o właściwościach funkcjonalnych oraz sposobach wykorzystania biblioteki *KBExpertLib*. Informacje o tej bibliotece nie były do tej pory nigdzie publiko-

wane, wcześniejsze publikacje anonsowały jedynie system *KBExplorator*. Autor ma nadzieję, że taka informacja będzie użyteczna dla osób zainteresowanych problematyką systemów z bazą wiedzy, jak również dla projektantów i programistów zainteresowanych wykorzystaniem baz wiedzy i wnioskowania w swoich systemach.

Drugim celem pracy jest przedstawienie wyników eksperymentów, których głównym zadaniem była ocena efektywności wnioskowania progresywnego dla baz wiedzy o różnej liczebności reguł – od 400 do ponad 20 tysięcy. Eksperymenty, przeprowadzone na komputerze o typowej konfiguracji sprzętowej, pozwalają jasno ocenić, jakich parametrów czasowych w zakresie wnioskowania można się spodziewać, gdy wykorzysta się bibliotekę *KBExpertLib*. Ocenie poddane zostały również czasy realizacji wybranych dodatkowych operacji oraz poziom zapotrzebowania na pamięć operacyjną.

Informacje przedstawione w niniejszej pracy mają pozwolić na wstępną ocenę użyteczności, łatwości wykorzystania i efektywności proponowanego rozwiązania, tuż przed jego udostępnieniem w postaci publicznego repozytorium kodu. Ograniczone ramy tego opracowania nie pozwalają jednocześnie na szczegółowe i rzetelne porównanie proponowanego rozwiązania z innymi, istniejącymi rozwiązaniami (przykładowo *CLIPS* [5], *JESS* [6]). Analiza porównawcza przewidywana jest jako kolejny etap badań.

2. Biblioteka *KBExpertLib*

Biblioteka *KBExpertLib* została od postaw zaprojektowana oraz zaimplementowana z wykorzystaniem języka Java i jest dedykowana dla systemów tworzonych z wykorzystaniem tego języka. Może być używana w aplikacjach klasy desktop – konsolowych i wykorzystujących graficzny interfejs użytkownika (GUI), realizowany z wykorzystaniem biblioteki *SWING* lub *JavaFX*. W przypadku aplikacji wykorzystujących GUI konieczne jest zaimplementowanie interfejsów programowych, pozwalających na realizację dialogu z użytkownikiem, w tym interfejsu wykorzystywanego w trakcie wnioskowania regresywnego. Biblioteka *KBExpertLib* składa się z czterech podstawowych pakietów: *kbcore, kbinfer, kbpartition, kbtool,* nie wymaga stosowania żadnych dodatkowych bibliotek poza biblioteką obsługi połączenia z bazą danych *MySQL* oraz obsługą raportowania *log4j* (The Apache Software Foundation).

Biblioteka *KBExpertLib* oferuje większość elementów i funkcji typowego, szkieletowego systemu ekspertowego. Głównym obszarem zastosowań biblioteki *KBExpertLib* jest realizacja dziedzinowych systemów ekspertowych – zarówno działających jako indywidualne programy, jak i stanowiących rozszerzenie systemów informatycznych o innym przeznaczeniu.

Biblioteka oferuje zestaw klas, pozwalających na: (i) reprezentowanie i manipulowanie regułowymi bazami wiedzy, (ii) realizację klasycznych wersji wnioskowania progresywnego i regresywnego, (iii) modularyzację regułowych baz wiedzy z wykorzystaniem koncepcji podziałów reguł [9], (iv) realizację zoptymalizowanego wnioskowania progresywnego i regresywnego, wykorzystującego modularne bazy wiedzy [9, 10]. Niniejsze opracowanie prezentuje główne właściwości biblioteki, szczegółowy opis formatu XML, oraz poszczególnych klas, zawiera dokumentacja techniczna, która będzie dostępna wraz z udostępnionym kodem. Tam też znaleźć będzie można szczegółowe diagramy UML, opisujące m.in. hierarchię proponowanych klas.

2.1. Obsługiwane bazy wiedzy

Biblioteka *KBExpertLib* wykorzystuje regułowe bazy wiedzy [9, 10], zakłada dwa sposoby fizycznej reprezentacji bazy wiedzy:

- Baza wiedzy może być utworzona przez aplikacje internetową *KBExplorator* [7]. Taka baza wiedzy jest przechowywana w wewnętrznej bazie danych systemu *KBExplorator*. Biblioteka KBExpertLib umożliwa pobranie zawartości bazy wiedzy z serwera, do czego wymagane jest posiadanie konta w systemie *KBExplorator* oraz istnienie w ramach danego konta odpowiedniej bazy wiedzy. System *KBExplorator* pozwala na import bazy i zapisanie jej w urządzeniu lokalnym w postaci dokumentu XML.
- Baza wiedzy może być zapisana w pliku XML przechowywanym w urządzeniu lokalnym. Taka baza wiedzy może być pobrana z systemu *KBExplorator* lub może być utworzona bez niego, do jej utworzenia i modyfikacji posłużyć się można dowolnym edytorem tekstowym, wymagana jest jedynie znajomość ustalonych znaczników, opisujących poszczególne elementy bazy wiedzy.

Niezależnie od źródła pozyskania bazy wiedzy, po jej pobraniu jest ona przechowywana w pamięci operacyjnej z wykorzystaniem obiektów klas zdefiniowanych w pakiecie *kbcore*. Należy zwrócić uwagę, że załadowana baza może być programowo modyfikowana. W szczególnym przypadku istnieje możliwość programowego utworzenia bazy przez programistę od podstaw, następnie może być ona zapisana w lokalnym pliku XML.

2.2. Wybrane właściwości funkcjonalne i przykłady wykorzystania

Pakiet *kbinfer* zawiera klasy oferujące możliwość realizacji klasycznego wnioskowania:

- Progresywnego (w przód), z możliwością ustalenia strategii wyboru reguł kandydujących oraz z możliwością ustalenia celu wnioskowania. Służy do tego klasa *KBForwardInferer*, Wnioskowanie to nie wymaga dodatkowych prac programistycznych, odbywa się w trybie

wsadowym. Dane wejściowe to zbiór reguł, zbiór faktów i opcjonalny cel, danymi wyjściowymi są nowe fakty oraz informacja o potwierdzeniu (lub nie) celu, o ile został określony. Do programowego manipulowania tymi elementami przeznaczona jest klasa *KBKnowledgeBase* oraz powiązane z nią związkami agregacji i kompozycji klasy *KBFacts*, *KBFact*, *KBLiteral*.

- Regresywnego (wstecz), z możliwością ustalenia strategii wyboru reguł kandydujących. Wykorzystanie wnioskowania regresywnego wymaga zastosowania dziedziczenia oraz zaimplementowania interfejsu wykorzystywanego w czasie wnioskowania – klasa *KBBackwardAbstractInferer* jest klasą abstrakcyjną. Implementowany interfejs jest odpowiedzialny za realizację dialogu z otoczeniem systemu, wykorzystywanego w przypadku, gdy nie można ustalić prawdziwości warunków reguł rozpatrywanych w czasie wnioskowania. Biblioteka oferuje gotową do wykorzystania klasę *KBBackwardConsoleInferer*, prowadzącą dialog z wykorzystaniem trybu konsolowego.
- Biblioteka oferuje również zoptymalizowane oraz nowe algorytmy wnioskowania bazujące na koncepcji podziałów reguł, wykorzystujących m.in. podziały decyzyjne oraz podziały bazujące na metodach grupowania [8-10]. Ze względu na specjalizowany charakter tej części biblioteki, i ograniczone ramy niniejszego opracowania, ten aspekt nie będzie szczegółowo opisywany.

Typowy scenariusz wykorzystania bibliotek *KBExpertLib* dla wnioskowania progresywnego może być następujący (wykorzystano operacje na poziomie standardowych strumieni wejścia-wyjścia). Utworzenie obiektu bazy wiedzy (klasa *KBKnowledeBase*), załadowanie zawartości bazy z serwera systemu KBExprorator (obiekt klasy *KBDataBaseLoader*):

```
KBKnowledgeBase base = new KBKnowledgeBase
KBDataBaseLoader kbLoader = new KBDataBaseLoader();
kbLoader.loadKnowledgeBase( "infekcje_wirusowe", base );
```

Dodanie faktów początkowych, niezbędnych dla wnioskowania w przód (metoda *addFactsFromText* klasy *KBKnowledeBase*), fakty są przechowywane tymczasowo:

```
base.addFactFromText( "kaszel", "=", "suchy" );
base.addFactFromText( "temperatura", ">=", "38.0" );
```

Wnioskowanie realizują odpowiednie klasy zdefiniowane w pakiecie *kbinfer*. Klasyczne wnioskowanie progresywne realizuje metoda *classicInference* klasy *KBForwardInferer*, parametrem metody jest stała określająca strategię doboru reguł, ze zbioru reguł dopasowanych do faktów (aktywowana będzie ostatnia znaleziona reguła).

```
KBForwardInferer infer = new KBForwardInferer( base );
infer.classicInference( KBInferer.RuleSelStrategy.LAST_RULE );
```

Po zakończeniu wnioskowania można sprawdzić, czy doprowadziło ono do wygenerowania nowych faktów, co odbywa się przez sprawdzenie stanu pola *newFactInfered* obiektu wnioskującego. Nowe fakty można wykorzystać w dowolny sposób, w podanym niżej przykładzie nowe fakty zapisywane są do pliku CSV w celu ich dalszego przetwarzania (wykorzystanie klasy *KBKnowledgeBaseTextWriter*):

```
if( infer.newFactInfered )
{
  KBKnowledgeBaseTextWriter writer = new KBKnowledgeBaseTextWriter( base );
  writer.factsToCSVFile( "nowefakty.csv" );
}
else
  System.out.println( "Brak nowych faktów" );
```

W przypadku wykorzystania bazy wiedzy zapisanej w pliku XML należy wykorzystać inną metodę klasy *KBDataBaseLoader*:

```
kbLoader.loadKnowledgeBaseFromXML( "infekcje_wirusowe.xml", base )
```

Typowy scenariusz wykorzystania wnioskowania wstecz może być następujący:

```
KBKnowledgeBase base = new KBKnowledgeBase();

KBDataBaseLoader kbLoader = new KBDataBaseLoader();

if( kbLoader.loadKnowledgeBaseFromXML("infekcje_wirusowe.xml", base ) )
{
    KBBackwardConsoleInferer infer = new KBBackwardConsoleInferer( base );
    // Utworznie celu dla wnioskowania wstecz
    KBLiteral goal = base.makeLiteralFromText( "choroba", "=", "grypa" );
    // Uruchomienie wnioskowania wstecz z ustalonym celem i domyślną strategią
    // wyboru reguły wśród reguła pasujących do celu wnioskowania
    if( infer.classicInferenceWithGoal( goal ) )
    {
        System.out.println( "Cel potwierdzony" );
        if( infer.newFactInfered )
            System.out.println( "Uzyskano nowe fakty w trakcie wnioskowania" );
    }
    else
        System.out.println( "Cel niepotwierdzony" );
}
```

Cel może również wystąpić dla wnioskowania progresywnego, ustalony uprzednio cel wykorzystywany jest jako dodatkowy parametr przy wywołaniu metody *classicInferenceWithGoal* obiektu klasy *KBForwardInferer*, ilustruje to poniższy przykład, tym razem aktywowana będzie pierwsza znaleziona reguła.

```
infer.classicInference( KBInferer.RuleSelStrategy.FIRST_RULE, goal );
```

Opisywana biblioteka pozwala również na wykorzystanie wielu nowych operacji, przykładem może być zbudowanie podziału decyzyjnego [9] – przedstawiony niżej przykład tworzy taki podział (dla uprzednio załadowanej bazy) i wyprowadza tekstowy opis grup do strumienia wyjściowego programu:

```
KBPartitioner divider = new KBPartitioner( base );
KBPartition partitions = divider.createDecisionPartition();
List<String>rulesGroups = KBPartitionText.getAsStringList( partitions );
for( String s : rulesGroups )
    System.out.println(s);
```

Podane przykłady wskazują, iż typowe scenariusze wykorzystujące elementy biblioteki, takie jak dostęp do baz wiedzy i wnioskowanie, nie wymagają realizacji złożonych operacji, pozwalając na uzyskanie efektów właściwych dla systemów ekspertowych z użyciem zaledwie kilku linii kodu. Następny podrozdział pracy poświęcony jest oszacowaniu wydajności opisywanych operacji dla baz o zróżnicowanym rozmiarze.

3. Eksperymenty

Celem eksperymentów opisywanych w niniejszej pracy jest ocena parametrów wydajnościowych wybranych operacji realizowanych przez bibliotekę *KBExpertLib*. Kluczowy wpływ na efektywność i zapotrzebowanie pamięciowe dla badanej biblioteki ma liczba reguł, eksperymenty zostały przeprowadzona z wykorzystaniem następujących baz wiedzy:

- *eval416* – baza wiedzy opracowana dla oceny skuteczności pracy przedstawicieli handlowych, liczy 416 reguł,
- *eval1199* – rozszerzona wersja bazy *eval416*, licząca 1199 reguł,
- *bud4438* – baza wiedzy wspomagająca podejmowanie decyzji w zakresie budownictwa, liczy *4438* reguł [11-12],
- *bud22190* – sztuczna baza wiedzy, powstała na bazie *bud4438* przez powielenie losowo wybranych i zmodyfikowanych reguł.

Tabela 1

Wyniki eksperymentów

Baza wiedzy	Średni czas odczytu [s]	Tworzenie podziałów [ms]	Czas wnioskowania [ms]		Rozmiar danych w pamięci [B]	
			W głąb	Wszerz	kbcore	kbpartition
eval416	0,312	11,34	81,23	41,53	95964	22538
eval1199	1,487	23,11	138,67	78,21	285316	24952
bud4438	27,39	112,87	186,69	372,36	1197148	372632
bud22190	670,83	224,55	934,35	301,46	4646348	382016

Testom podlegała biblioteka w wersji dedykowanej dla aplikacji typu desktop. Jak wcześniej zaznaczono, badania dla innych platform stanowią przedmiot osobno realizowanych badań. Do opisywanych eksperymentów wybrano operacje, mające istotny wpływ na efektyw-

ność operacji oraz obciążenie pamięci operacyjnej. W drugim etapie dokonano oceny efektywności wnioskowania progresywnego dla dwóch wersji algorytmu.

Dla każdego przebiegu testowego wykonywano pomiary czasu trzema alternatywnymi metodami, każdy test powtarzano dziesięć razy, wyniki uśredniano. Środowiskiem testowym był typowy komputer klasy PC (Intel i5 2.5 GHz, 16 GB RAM, klasyczny dysk mechaniczny, Windows 10, 64-bitowy). Wyniki badań zostały przedstawione w tabeli 1. Czasy odczytu bazy i utworzenia jej pamięciowej reprezentacji dla baz liczących 416 i 1199 są zadowalające, oscylują w przedziale 0,3-1,5 sek. Dla bazy liczącej 4438 czas odczytu wzrósł do blisko 30 sek., natomiast odczyt dla bazy liczącej 22190 reguł zajął już ponad 11 min. Dla dużych baz ich czas odczytu z pliku XML może być nieakceptowalny. Eksperymenty dotyczące wnioskowania w przód wykazały zadowalającą efektywność opracowanych algorytmów, mimo iż w tej wersji biblioteki nie zaimplementowano algorytmu RETE [6], czasy realizacji wnioskowania dla bazy liczącej 22190 reguł nie przekroczyły 1 sekundy. Czas tworzenia podziałów decyzyjnych dla największych baz nie przekroczył 0,2 sek., wydaje się zatem, że generowanie podziału decyzyjnego, stanowiącego podstawę dla optymalizacji algorytmów wnioskowania [8-10], jest procesem o zadowalającej efektywności. Eksperymenty potwierdziły również liniową zależność czasu budowania podziału decyzyjnego względem liczby reguł.

4. Podsumowanie i wnioski końcowe

Celem pracy było przedstawienie informacji pozwalających na wstępną ocenę użyteczności, metod wykorzystania i efektywności biblioteki *KBExpertLib*, tuż przed udostępnieniem w postaci publicznego repozytorium kodu. Przeprowadzone eksperymenty wykazały, iż czasy realizacji podstawowych operacji są akceptowalne z punktu widzenia klasycznej aplikacji klasy desktop działającej w trybie interaktywnym. Nawet dla największych baz czasy wnioskowania nie przekraczały 1 sekundy. Zatem w przypadku uaktywniania pojedynczych przebiegów wnioskowania można się spodziewać zadowalających czasów odpowiedzi systemu. Otwartym zagadnieniem jest problem oceny efektywności wnioskowania w przypadku uaktywniania serii wnioskowań, np. w przypadku systemu działającego wsadowo i obsługującego wiele żądań jednocześnie. Dotyczy to również wnioskowania jako usługi sieciowej w systemach WWW oraz systemów wbudowanych. W przypadku tych ostatnich interesująca jest również analiza zapotrzebowania na zasoby w warunkach wykorzystywania urządzeń o ograniczonej mocy obliczeniowej i zasobach. Te zagadnienia są m.in. przedmiotem zainicjowanych, lecz nieukończonych jeszcze prac eksperymentalnych.

Istotnym problemem implementacyjnym jest nieakceptowaly czas ładowania dużych baz wiedzy z plików XML. Należy rozważyć opracowanie formatu pliku bazy wiedzy w zoptymalizowanej postaci binarnej, pozwalającej na szybki odczyt bazy bez konieczności wykonywania analiz przez parser XML. W sensie realizacyjnym zakłada się wykorzystanie mechanizmu serializacji oraz alternatywnie, bezpośredniego wykorzystania plików binarnych.

Podsumowując, badania wydajnościowe biblioteki KBExpertLib wykazały zadowalającą efektywność podstawowych operacji dla aplikacji klasy desktop. Badania będą kontynuowane dla wersji biblioteki pracującej na serwerze WWW oraz wersji dedykowanej dla urządzeń mobilnych. Następny etap badań przewiduje analizę porównawczą z rozwiązaniami alternatywnymi – pakietami *CLIPS* i *JESS*.

BIBLIOGRAFIA

1. Grzymala-Busse J.W.: Managing uncertainty in expert systems. Springer Science & Business Media, Vol. 143, 1991.
2. Skowron A., Komorowski J., Pawlak Z., Polkowski L.: Handbook of data mining and knowledge discovery. Oxford University Press, Inc., New York, NY, USA 2002.
3. Bazan J., Skowron A., Synak P.: Discovery of decision rules from experimental data. Proceedings of the Third International Workshop on Rough Sets and Soft Computing, 1994, s. 526÷533.
4. Stefanowski J., Vanderpooten D.: Induction of decision rules in classification and discovery-oriented perspectives. International Journal of Intelligent Systems, Vol. 16(1), 2001, s. 13÷27.
5. CLIPS: A Tool for Building Expert Systems, dostępne online: http://clipsrules. sourceforge.net.
6. JESS Information, dostępne online: http://herzberg.ca.sandia.gov.
7. Nowak-Brzezińska A.: KbExplorator a inne narzędzia eksploracji regułowych baz wiedzy. Studia Informatica, Vol. 36, No. 1, Gliwice 2015, s. 215÷226.
8. Nowak-Brzezińska A., Simiński R.: Knowledge mining approach for optimization of inference processes in rule knowledge bases. Lecture Notes in Computer Science, Vol. 7567, Springer-Verlag, 2012, s. 19÷27.
9. Simiński R.: Extraction of Rules Dependencies for Optimization of Backward Inference Algorithm. Beyond Databases, Architectures, and Structures, Communications in Computer and Information Science, Vol. 424, Springer International Publishing, Springer-Verlag, 2014, s. 191÷200.

10. Nowak-Brzezińska A., Simiński R.: New inference algorithms based on rules partition. CS&P 2014, Informatik-Berichte, Vol. 245, Humboldt-University, Chemnitz, Germany 2014.
11. Simiński R., Manaj M.: Implementation of expert subsystem in the web application – selected practical issues. Studia Informatica , Vol. 36, No. 1, 2015, s. 131÷143.
12. Jach T., Xieski T.: Inference in expert systems using natural language processing. [in:] Kozielski S., Mrozek D., Kasprowski P., Małysiak-Mrozek B., Kostrzewa D. (eds.): Beyond Databases, Architectures and Structures. Communications in Computer and Information Science, Vol. 521, Springer International Publishing, 2015, s. 288÷298.
13. Nowak-Brzezińska A., Rybotycki T.: Visualization of medical rule-based knowledge bases. Journal of Medical Informatics & Technologies, Vol. 24, 2015, s. 91÷98.

Abstract

This article describes the short functional description of the *KBExpertLib* software package and the performance study of basic functions provided by the library. The *KBExpertLib* is the own, originally designed software package which provides most of the expert system shell's common functions. This library allows to build expert systems using Java programming language. In the first part of this paper the structure of the library and example of its utilization is presented. The second part present results of the experimental research focused on the time efficiency of the basic *KBExpertLib* operations. The estimation of the memory occupation of the library data structures are also presented..

Adres

Roman SIMIŃSKI: Uniwersytet Śląski, Instytut Informatyki, ul. Będzińska 29, 41-200 Sosnowiec, Polska, roman.siminski@us.edu.pl.