Uniwersytet Ekonomiczny w Katowicach

Studia Ekonomiczne. Zeszyty Naukowe
Uniwersytetu Ekonomicznego w Katowicach
ISSN 2083-8611 Nr 342 · 2017

Informatyka i Ekonometria 11

Paweł Ziemba

Akademia Jakuba z Paradyża
w Gorzowie Wielkopolskim
Wydział Techniczny
pziemba@ajp.edu.pl

Agata Ziemba

Uniwersytet Szczeciński
Wydział Humanistyczny
agataziemba16@gmail.com

DOBÓR EDYTORA NA POTRZEBY BUDOWY ONTOLOGICZNEJ BAZY WIEDZY

Streszczenie: Sprawne zarządzanie wiedzą pozwala uniknąć ponownego poszukiwania rozwiązań problemów, które już wcześniej zostały rozwiązane. Dlatego też wiedzę należy przechowywać w bazach wiedzy, które obecnie są najczęściej konstruowane w postaci ontologicznej. Istotnymi zagadnieniami, związanymi z budową baz wiedzy, są: dobór języka reprezentacji wiedzy, metodyki budowy bazy oraz wybór narzędzia wspierającego taką budowę, tj. edytora ontologii. Celem artykułu jest dobór edytora ontologii w oparciu o metody wielokryterialnej analizy decyzyjnej. Zastosowanie znalazły tutaj metody: wydzielania dla minimalnej wartości atrybutu oraz Fuzzy TOPSIS. W artykule rozpatrzono 8 współczesnych edytorów ontologii. W efekcie zastosowania dwuiteracyjnej procedury wielokryterialnej wskazano edytor Protege jako ten, który charakteryzuje się optymalnym poziomem poszczególnych cech, niezbędnych w tego rodzaju oprogramowaniu.

Słowa kluczowe: baza wiedzy, ontologia, edytor ontologii, wielokryterialna analiza decyzyjna, Fuzzy TOPSIS.

JEL Classification: C44, C88.

Wprowadzenie

Wiedzę ogólnie definiuje się jako zasób wiadomości z jakiejś dziedziny, natomiast praktycy ujmują ją jako informację, która została zrozumiana, wzbogacona o osąd i wykorzystana w działaniu [Błaszczuk i in., 2004, s. 13]. Jeżeli chodzi o cechy wiedzy, to: może być ona tworzona różnymi metodami, jest trudna do uchwycenia i pełnego wykorzystania, jest względna, wieloznaczna i dynamiczna, może się dezaktualizować, obniża poziom niepewności w trudnych przedsięwzięciach obarczonych ryzykiem [Mikuła, Pietruszka-Ortyl, Po-

tocki, 2002, s. 73]. Pod postacią wiedzy mogą być gromadzone wszelkie informacje o rozwiązanych już problemach, zbadanych obszarach lub wykonanych badaniach. Dlatego też sprawne nią zarządzanie pozwala uniknąć wielokrotnego poszukiwania rozwiązań określonych problemów oraz wykonywania tych samych badań. Formalnie zarządzanie wiedzą definiowane jest jako proces, za pomocą którego organizacja generuje bogactwo na bazie swoich intelektualnych lub opartych na wiedzy aktywów organizacyjnych [Bukowitz, Williams, 1999, s. 2]. Zarządzanie wiedzą określa procesy lokalizowania wiedzy, jej pozyskania, wzbogacania, dzielenia się nią i rozpowszechniania, wykorzystania i gromadzenia [Adamczewski, 2006].

Postęp w rozwoju systemów informacyjnych sugeruje użycie ontologii jako podstawowego narzędzia zarządzania wiedzą [Segev, Gal, 2008]. Wynika to z faktu, że przedstawienie wiedzy z wykorzystaniem ontologii zapewnia jej formalną i uporządkowaną reprezentację [Haghighi i in., 2013]. Formalna postać ontologii daje ramy dla skutecznego i wydajnego współdzielenia wiedzy oraz jej ponownego wykorzystania [Guzman-Arenas, Cuevas, 2010]. Dodatkowo zastosowanie ontologii pozwala przeprowadzać wnioskowanie, wywodząc nową wiedzę z tej, która została zapisana w ontologii pełniącej rolę bazy wiedzy [Chen i in., 2012].

Każda ontologia powinna być wyrażona za pomocą określonego języka reprezentacji wiedzy, a do jej budowy można posłużyć się odpowiednim edytorem ontologii. Ponadto przy budowie ontologii należy określić metodykę, według której będzie ona powstawała. W pracach [Ziemba, Jankowski, Wolski, 2015; Wątróbski, Ziemba, 2015] przedstawiono zagadnienia doboru języka reprezentacji oraz metodyki budowy ontologii. W niniejszym artykule za cel postawiono natomiast wybór edytora, oferującego największe możliwości w zakresie konstrukcji ontologii. W rozdziale 1 przedstawiono podstawowe zagadnienia związane ze stosowalnością ontologii do reprezentacji wiedzy. Rozdział 2 zawiera omówienie procedury badawczej i zastosowanych w niej metod. W kolejnych rozdziałach scharakteryzowano poszczególne edytory ontologii oraz przedstawiono przebieg procesu wyboru edytora. Artykuł kończy się wnioskami z przeprowadzonego badania.

1. Ontologia jako narzędzie do reprezentacji wiedzy

Wiedza jest podstawą sztucznej inteligencji, a więc kwestie związane z jej reprezentacją, zrozumieniem, projektowaniem i implementacją są bardzo istotne dla wszelkich systemów inteligentnych [Chaudhary i in., 2012]. O reprezentacji

wiedzy mówi się ogólnie w kontekście jej przedstawiania w taki sposób, który umożliwia jej automatyczne przetwarzanie przez komputer [Hartley, 1985]. W sztucznej inteligencji przez reprezentację wiedzy rozumie się natomiast połączenie struktur danych oraz procedur interpretacyjnych operujących na tych danych [Rao, Jain, 1988]. Procedury te – właściwie użyte – powinny prowadzić do inteligentnego zachowania systemu. Oznacza to, że system taki powinien potrafić przeprowadzać wnioskowanie w sposób zbliżony do człowieka [Abraham, 2003].

Stosowalność ontologii do reprezentacji wiedzy wynika z samej definicji ontologii, która mówi, że ontologia jest „jednoznaczną, formalną specyfikacją współdzielonej konceptualizacji" [Gruber, 1993, s. 199]. „Jednoznaczna" i „formalna" oznaczają, że zawartość ontologii jest opisana za pomocą mechanizmów formalnych i logiki matematycznej. „Specyfikacja konceptualizacji" odnosi się do tego, że ontologia jest uproszczonym modelem jakiejś części wszechświata reprezentującym wiedzę dziedzinową. Z kolei „współdzielenie" wskazuje na to, że ontologia odzwierciedla powszechne rozumienie modelowanej dziedziny [Ivanović, Budimac, 2014]. Podobna definicja mówi o tym, że w informatyce ontologia jest traktowana jako struktura danych i narzędzie do reprezentacji danych, pozwalające współdzielić i ponownie wykorzystywać wiedzę w systemach sztucznej inteligencji korzystających ze wspólnego słownictwa [Guzman-Arenas, Cuevas, 2010]. Definicje te odnoszą się do faktu, że za pomocą ontologii można przedstawić wiedzę dziedzinową w postaci zrozumiałej dla maszyn i ludzi, a następnie tę wiedzę współdzielić i wielokrotnie wykorzystywać. Bardziej złożone ontologie, poza reprezentowaniem i modelowaniem wiedzy, potrafią także wnioskować nową wiedzę z tej, która została zapisana w bazie wiedzy [Valaski, Malucelli, Reinehr, 2012].

Zastosowanie odpowiedniego edytora do budowy ontologii może znacząco ułatwić i przyspieszyć proces jej implementacji. Wśród oprogramowania związanego z zarządzaniem wiedzą pojawia się coraz więcej edytorów ontologii, których dystrybucja odbywa się zarówno na zasadach open source, jak również w oparciu o licencje komercyjne. Problemem zaś staje się wybór takiego spośród nich, który będzie optymalny przy realizacji bazy wiedzy z określonej dziedziny [Dudycz, 2015].

2. Procedura badawcza

Opierając się na publikacji Dudycz [2015], przyjęto, że problem wyboru edytora do budowy ontologicznej bazy wiedzy jest problemem wielokryterialnym. Dlatego też przyjęta procedura badawcza jest ściśle związana z przebiegiem procesu decyzyjnego, stosowanego w wielokryterialnej analizie decyzyjnej. Proces decyzyjny jest zwykle złożony z czterech etapów, którymi są:

(I) strukturalizacja problemu decyzyjnego – zdefiniowanie celu, problematyki i wariantów decyzyjnych;
(II) artykułowanie i modelowanie preferencji – opracowanie zbioru kryteriów;
(III) agregacja ocen wariantów (preferencji);
(IV) rekomendacja [Roy, 1996; Guitouni, Martel, 1998].

Roy zaznacza przy tym, że etapy te nie przebiegają szeregowo, lecz np. pewne elementy etapu I mogą wymagać wykonania elementów etapu II. Podobnie, proces decyzyjny nie może zostać uproszczony poprzez wyeliminowanie poszczególnych etapów. W etapie I Roy [1996] wyróżnia 4 problematyki decyzyjne:

– α – problematyka wyboru małej liczby „dobrych" wariantów;
– β – problematyka sortowania wariantów do określonych klas;
– γ – problematyka rankingu, czyli ułożenia wariantów w porządku od najlepszego do najgorszego;
– δ – problematyka opisu wydajności wariantów pod względem kryteriów.

Natomiast w etapie III zdefiniowane są trzy możliwe podejścia operacyjne decydenta w zakresie agregacji wydajności wariantów: (1) użycie pojedynczego zsyntetyzowanego kryterium bez nieporównywalności; (2) relacja przewyższania, dopuszczająca nieporównywalność wariantów; (3) interaktywne osądy w kolejnych iteracjach przybliżające do rozwiązania. Etap IV wymaga w szczególności wyboru procedury obliczeniowej (metody MCDA, ang. *Multi-Criteria Decision Analysis method*), zależnie od problematyki decyzji i podejścia operacyjnego decydenta [Roy, 1996]. Przyjęta procedura badawcza jest w istocie kombinacją różnych problematyk decyzyjnych i podejść operacyjnych.

Na wstępie realizacji procedury badawczej dokonano przeglądu możliwych do wyboru wariantów decyzyjnych. Ze względu na ich liczność przyjęto, że wybór zostanie dokonany iteracyjnie. Mianowicie w pierwszej iteracji procedury postanowiono rozpatrzyć problematykę wyboru α i ograniczyć liczbę wariantów, wybierając najlepsze do dalszego rozpatrzenia. Do wyboru małej liczby dobrych wariantów zastosowano metodę wydzielania dla minimalnej wartości atrybutu (ang. *Conjunctive method*, *Satisficing method*). W metodzie tej do dalszego rozpatrzenia wybiera się warianty spełniające w zadowalającym stopniu poszczególne kryteria, odrzucając pozostałe [Hwang, Yoon, 1981]. W drugiej iteracji rozpatrzono warianty, które nie zostały odrzucone w iteracji pierwszej, tym razem rozpatrując problematykę rankingu γ, agregując warianty do pojedynczego zsyntetyzowanego kryterium. Ze względu na możliwość występowania niepewności ocen zastosowano tutaj rozmyte oceny lingwistyczne i metodę Fuzzy TOPSIS [Chen, 2000; Chen, Lin, Huang, 2006].

W metodzie Fuzzy TOPSIS ocena *i*-tego wariantu względem *j*-tego kryterium wyrażana jest jako trapezoidalna liczba rozmyta $\dot{x}_{ij}=(a_{ij},b_{ij},c_{ij},d_{ij})$, natomiast waga *j*-tego kryterium jest liczbą rozmytą $\dot{w}_j=(w_{j1},w_{j2},w_{j3},w_{j4})$. Zbiór kryteriów może być podzielony na kryteria typu zysk i koszt. Dla kryteriów typu zysk znormalizowane wartości rozmyte uzyskuje się według wzoru:

$$\dot{r}_{ij} = \left(\frac{a_{ij}}{d_j^*}, \frac{b_{ij}}{d_j^*}, \frac{c_{ij}}{d_j^*}, \frac{d_{ij}}{d_j^*}\right),$$

gdzie $d_j^* = \max_i d_{ij}$.

Natomiast dla kryteriów kosztowych normalizacja przebiega według wzoru:

$$\dot{r}_{ij} = \left(\frac{a_j^-}{d_{ij}}, \frac{a_j^-}{c_{ij}}, \frac{a_j^-}{b_{ij}}, \frac{a_j^-}{a_{ij}}\right),$$

gdzie $a_j^- = \min_i a_{ij}$.

Po uwzględnieniu wag kryteriów uzyskuje się ważone znormalizowane wartości wariantów $\tilde{v}_{ij}=\dot{r}_{ij}(\bullet)\dot{w}_j$. Na podstawie ważonych znormalizowanych wartości wariantów wyznaczane są: rozmyte rozwiązanie idealne $A^* = (\tilde{v}_1^*, \tilde{v}_2^*, \dots, \tilde{v}_n^*)$ oraz rozmyte rozwiązanie anty-idealne $A^- = (\tilde{v}_1^-, \tilde{v}_2^-, \dots, \tilde{v}_n^-)$, gdzie $\tilde{v}_j^* = \max_i\{\tilde{v}_{ij4}\}$ i $\tilde{v}_j^- = \min_i\{\tilde{v}_{ij1}\}$. Odległość *i*-tego wariantu od rozwiązania idealnego (d_i^*) i anty-idealnego (d_i^-) wyznaczana jest według wzorów:

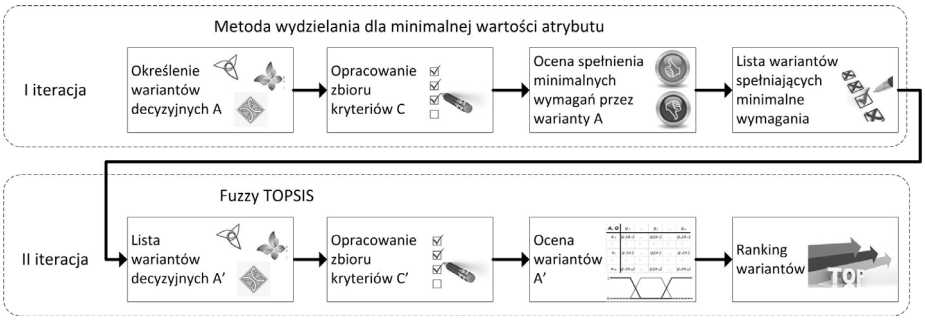
$$d_i^* = \sum_{j=1}^n d_v\left(\tilde{v}_{ij}, \tilde{v}_j^*\right),$$
$$d_i^- = \sum_{j=1}^n d_v\left(\tilde{v}_{ij}, \tilde{v}_j^-\right),$$

gdzie $d_v(\bullet,\bullet)$ jest miarą odległości między dwoma liczbami rozmytymi.

Globalna ocena każdego wariantu wyrażana jest przez współczynnik odległości od rozmytego rozwiązania idealnego i anty-idealnego. Współczynnik ten opisany jest wzorem:

$$CC_i = \frac{d_i^-}{d_i^* + d_i^-}.$$

Schemat przebiegu przyjętej procedury badawczej przedstawiono na rys. 1.



Rys. 1. Przyjęta procedura badawcza wyboru edytora ontologii

Źródło: Opracowanie własne.

3. Przegląd edytorów ontologii

Edytory są narzędziami programowymi, opartymi na formalizmie ontologii, które umożliwiają interaktywną budowę i modyfikację ontologii za pomocą graficznego interfejsu użytkownika. Edytory ontologii zaliczają się do szerszej grupy narzędzi inżynierii ontologii, a można je określić również jako narzędzia rozwoju ontologii. Niektóre z edytorów są przeznaczone do budowy ontologii dla konkretnej dziedziny (np. OBO-Edit i DAG-Edit przeznaczone dla bioinformatyki), a inne z nich wspierają budowę ontologii z wykorzystaniem różnych technik, jak np. przetwarzanie języka naturalnego i uczenie maszynowe [Mizoguchi, Kozaki, 2009]. Wśród klasycznych edytorów ontologii, których podstawowym zadaniem jest umożliwienie interaktywnej budowy ontologii, wyróżnić można edytory działające na stronach WWW (np. WebProtege; www 1) oraz narzędzia instalowane lokalnie w systemie operacyjnym. W niniejszej pracy dokonano analizy edytorów służących przede wszystkim do budowy i edycji ontologii, które działają lokalnie w systemie. Pominięto natomiast narzędzia, w których praca odbywa się tylko za pośrednictwem przeglądarki internetowej i strony WWW. Nie uwzględniono tutaj również edytorów ontologii dla konkretnych dziedzin. Jednak, nawet przy tak ograniczonej analizie, należałoby uwzględnić bardzo dużo narzędzi do budowy i edycji ontologii. Wobec tego w niniejszym badaniu zdecydowano się ująć tylko najpopularniejsze edytory, tj.: Protege, Hozo, NeOn Toolkit, KAON2, DogmaModeler, OntoStudio i TopBraid Composer.

Protege jest bezpłatnym edytorem, opartym na licencji open source. Udostępnia on intuicyjny i łatwy w użyciu interfejs graficzny do opracowywania ontologii. Protege napisany jest w języku JAVA, w związku z czym charakteryzuje się przenośnością oraz niezależnością od platformy sprzętowej i programowej. Umożliwia on pracę grupową w architekturze klient–serwer. Ponadto dostępnych jest do niego wiele rozszerzeń, znacznie zwiększających jego funkcjonalność w zakresie m.in.: wizualizacji ontologii, wnioskowania, zarządzania wersjami ontologii, edycji reguł SWRL i tworzenia zapytań SPARQL. Protege zawiera również szereg narzędzi służących do refaktoryzacji ontologii, które pozwalają na: automatyczną zamianę identyfikatorów URI, rozłączanie i łączenie aksjomatów opisujących klasy, kopiowanie, usuwanie i przenoszenie aksjomatów oraz łączenie ontologii. Obsługuje on język OWL 2, m.in. w notacjach RDF, OWL XML i Manchester. Niewątpliwą zaletą Protege jest bardzo duża społeczność jego użytkowników, w związku z czym dostępnych jest wiele źródeł informacji na temat tego edytora. Wobec tego uzyskanie pomocy w zakresie jego

wykorzystania nie stanowi problemu. Wraz z edytorem Protege dostarczana jest też bardzo bogata dokumentacja, umożliwiająca szybką naukę obsługi tego edytora i prezentująca sposoby tworzenia w nim zaawansowanych konstrukcji ontologicznych. Edytor ten jest stale rozwijany pod względem nowych funkcjonalności i opracowywane są dla niego nowe rozszerzenia [Gomez-Perez, Ruiz, 2010; Casellas, 2011; Alatrish, 2012; www 2].

Hozo jest dostępny bezpłatnie na zasadach licencji open source, a ponadto dostępne jest również tzw. Hozo API, pozwalające wykorzystywać dane utworzone w edytorze Hozo w aplikacjach Java. Hozo dostarcza interfejs graficzny, pozwalający przeglądać i modyfikować ontologie za pomocą operacji myszy. Hozo jest oparty na teorii ontologii i metodologiach budowy ontologii, a budowane w nim ontologie są reprezentowane za pomocą języka ram opartego na XML. Edytor Hozo jest zaimplementowany w języku JAVA. Udostępnia on wizualizację ontologii oraz szereg narzędzi służących do wspólnego opracowywania ontologii (zarządzanie wersjami, współdzielenie ontologii, pracę grupową wraz z unikaniem konfliktów podczas modyfikacji ontologii przez użytkowników). Mechanizm wnioskujący Hozo na bieżąco weryfikuje spójność ontologii. Edytor ten wspiera języki: RDF(S) i OWL. Jest dla niego dostępna względnie bogata dokumentacja, jednak jest to produkt wywodzący się z Japonii, wobec czego społeczność jego anglojęzycznych użytkowników nie jest zbyt duża. Może to skutkować trudnościami w uzyskaniu pomocy od społeczności użytkowników podczas pracy z tym edytorem [Mizoguchi, Kozaki, 2009; Casellas, 2011; www 3].

NeOn Toolkit jest dostępny w dwóch wersjach. Wersja podstawowa udostępniona jest na licencji Eclipse Public License i jest ona bezpłatna, a wersja rozszerzona oferuje dodatkowe możliwości i wykorzystuje licencję komercyjną. Edytor ten jest oparty na języku JAVA, a w szczególności na środowisku programistycznym Eclipse, w związku z czym jego interfejs użytkownika jest zbliżony właśnie do Eclipse. Wśród funkcji edytora należy wymienić wizualizację ontologii, odpytywanie ontologii za pomocą zapytań SPARQL i wnioskowanie za pomocą mechanizmów Pellet i Hermit. Dla tego edytora dostępne są również liczne rozszerzenia związane m.in. z: tworzeniem dokumentacji ontologii, oceną ontologii, pozyskiwaniem wiedzy, dopasowywaniem ontologii, obsługą języka FLogic, czy też wersjonowaniem i pracą grupową. Problemem jest jednak to, że część rozszerzeń dostępna jest na licencjach komercyjnych, w związku z czym trzeba ponosić opłaty za korzystanie z nich. NeOn Toolkit wspiera edycję ontologii z wykorzystaniem języków FLogic, RDF(S) i OWL2. Jeżeli chodzi o język OWL, to NeOn Toolkit umożliwia jego stosowanie w notacjach RDF, OWLX

i Manchester. Społeczność jego użytkowników jest względnie liczna, a do edytora dostępna jest stosunkowo obszerna dokumentacja [Gomez-Perez, Ruiz, 2010; Casellas, 2011; www 4].

KAON2 jest dostępny bezpłatnie do badań akademickich, ale istnieje też komercyjna wersja tego edytora (OntoBroken OWL). Oparty jest on na językach OWL-DL i FLogic. Podobnie jak wcześniej omówione edytory, KAON2 zaimplementowany jest w języku JAVA, a umożliwia generowanie instancji w ontologii na podstawie baz relacyjnych oraz tworzenie zapytań SPARQL. Wykorzystany w nim mechanizm wnioskujący, w przeciwieństwie do większości innych narzędzi wnioskowania, nie wykorzystuje algorytmów tablicowych, lecz redukuje dialekt SHIQ(D) do programu datalog. Wadą tego edytora jest fakt występowania problemów przy stosowaniu ograniczeń w ontologii. Ponadto ograniczenie ekspresywności ontologii tylko do dialektu SHIQ(D) logiki opisowej powoduje, że nie obsługuje on np. konceptów wyliczalnych, a także rozłączności, symetrii i zwrotności ról. Obsługuje on języki OWL-DL (w notacjach OWL XML i RDF), FLogic i język reguł SWRL. Społeczność jego użytkowników nie jest duża, a dokumentacja dostarczana wraz z edytorem jest bardzo uboga [Casellas, 2011; www 5].

DogmaModeler jest edytorem dostępnym na licencji GPL. Oparty jest on na metodzie modelowania z wykorzystaniem graficznych diagramów ORM (ang. *Object Role Modeling*), co ma ułatwiać proces modelowania zwykłym użytkownikom. Podobnie jak znaczna większość innych edytorów, oparty on jest na języku JAVA. W związku z wykorzystaniem diagramów ORM budowane ontologie są wizualizowane, a ponadto edytor wspiera werbalizację diagramów OML z użyciem języka „pseudo-naturalnego" i wnioskowanie przy użyciu mechanizmu Racer. DogmaModeler obsługuje import i eksport plików w językach: OWL 2, RDF i RDFS oraz w dialekcie SHIQ logiki opisowej. Społeczność użytkowników tego edytora jest bardzo mała, a dostępna pomoc i dokumentacja nie są zbyt obszerne [Casellas, 2011; www 6].

OntoStudio jest komercyjną kontynuacją edytora OntoEdit, który nie jest już dostępny w sieci Internet. Model danych edytora OntoStudio, tak jak było to w przypadku OntoEdit, bazuje na ramach. Podobnie jak NeOn Toolkit, edytor ten jest oparty na języku JAVA, a w szczególności na środowisku programistycznym Eclipse. W zasadzie NeOn Toolkit i OntoStudio bazują na tych samych kodach źródłowych [www 4]. Funkcje oferowane przez ten edytor to np.: grupowe opracowywanie i współdzielenie ontologii, wizualizacja ontologii, eksport ontologii do modelu relacyjnego bazy danych oraz import danych m.in. z baz relacyjnych, formatu Excel, UML 2.0. Obsługiwane przez OntoStudio języki to: OWL, RDF(S), logika obiektowa, język regułowy RIF oraz język zapy-

tań SPARQL. OntoStudio wykorzystuje mechanizm wnioskujący OntoBroken, będący komercyjnym rozszerzeniem mechanizmu zawartego w edytorze KAON2. Ze względu na fakt, że OntoStudio jest edytorem komercyjnym, dostępne jest do niego wsparcie producenta oraz bogata dokumentacja [Escorcio, Cardoso, 2007; Alatrish, 2012; www 7].

TopBraid Composer jest oprogramowaniem komercyjnym (Standard Edition i Maestro Edition), ale dostępna jest również jego bezpłatna wersja o ograniczonej funkcjonalności (Free Edition). Edytor ten ma swoje korzenie w edytorze Protege, w związku z czym niektóre jego elementy są podobne do tych, które zastosowano w Protege. Interfejs graficzny edytora Top Braid Composer jest oparty na środowisku programistycznym Eclipse i Jena API. Jego bezpłatna wersja umożliwia edycję ontologii i wspiera zapytania SPARQL, ale nie zawiera chociażby: wizualizacji ontologii, wsparcia refaktoryzacji, rozbudowanej obsługi zapytań SPARQL i konwersji do wielu formatów bazodanowych. Dla wersji darmowej brak jest również wsparcia producenta [Gomez-Perez, Ruiz, 2010; Alatrish, 2012; www 8].

Poza scharakteryzowanymi edytorami ontologii dostępne są także takie edytory, jak np.: TwoUse, mający formę rozszerzenia dla środowiska Eclipse; przeznaczony do pracy grupowej Knoodl; Vitro, powstały w ramach projektu VIVO i inne. Również wiele edytorów przestało być rozwijanych i nie są one już dostępne. Przykładem takich narzędzi są chociażby: Ontolingua, Chimaera, WebODE i OntoEdit. Istnieją także edytory, których rozwój zakończył się ponad dekadę temu, ale nadal są dostępne na oficjalnych stronach producentów, np. Apollo i Swoop. Podsumowanie przytoczonych wcześniej informacji na temat podstawowych cech poszczególnych edytorów ontologii zostało zawarte w tab. 1.

Tabela 1. Podstawowe cechy wybranych edytorów ontologii

Cecha	**Protege**	**Hozo**	**NeOn Toolkit**	**KAON2**	**Dogma Modeler**	**Onto Studio**	**TopBraid Composer Free Edition**	**TopBraid Composer Standard/ Maestro Edition**
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
Wersja	5.2.0	5.2.36	2.5.2	–	–	3.2.5	5.2.2	5.2.2
Licencja użytkowania	Open Source	Open Source	Eclipse Public License	Darmowa	GPL Open Source	Komercyjna	Darmowa	Komercyjna
Języki ontologii	RDF, OWL, OWL 2	RDF(S), OWL	FLogic (plug-in), RDF, OWL, OWL 2-częściowo	FLogic, OWL	RDF, RDFS, DL-SHIQ, OWL 2	RDF(S), OWL, Object Logic (FLogic)	RDFS, OWL, OWL 2	RDF, RDFS, OWL, OWL 2

cd. tabeli 1

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
Reguły	+	–	+	+	–	+	+	+
SPARQL	+	–	+	+	–	+	+	+
Wnioskowanie	+	+	+	+	+	+	+	+
Wizualizacja	+	+	+	+	+	+	–	+
Integracja	+	+	+	–	+	+	+	+
Wersjonowanie	+	+	+	–	–	+	+	+
Praca grupowa	+	+	+	+	–	+	+	+
Modularność (rozszerzenia plug-in)	+	–	+	–	–	+	+	+
Dokumentacja	+	+	+	–	+	+	+	+
Wsparcie	+	+	+	–	–	+	–	+
Data wydania	3.2017	9.2011	12.2011	6.2008	5.2012	2.2017	1.2017	1.2017

Źródło: Opracowanie własne na podstawie: Escorcio, Cardoso [2007]; Mizoguchi, Kozaki [2009]; Kotis, Vouros [2010]; Gomez-Perez, Ruiz [2010]; Kapoor, Sharma [2010]; Mishra, Kumar [2011]; Casellas [2011]; Alatrish [2012]; [www 2]; [www 3]; [www 4]; [www 5]; [www 6]; [www 7]; [www 8].

W tab. 1 zawarto informacje na temat licencjonowania edytorów i wsparcia przez nie języków: ontologii, reguł (najczęściej SWRL) i zapytań SPARQL. Tab. 1 uwzględnia również: obecność mechanizmów wnioskujących w poszczególnych edytorach, dostępność narzędzi do wizualizacji ontologii oraz umożliwianie przez edytory integracji i zarządzania wersjami ontologii, a także pracę grupową nad ontologią i możliwość dodawania do edytora dodatkowych funkcji w postaci rozszerzeń plug-in. Dodatkowo wskazano na istnienie obszernej dokumentacji do poszczególnych edytorów i łatwość uzyskania wsparcia producenta lub użytkowników danego edytora. Ponadto poprzez datę wydania ostatniej wersji edytora zasygnalizowano, czy producent danego narzędzia nadal je rozwija.

4. Dobór edytora ontologii

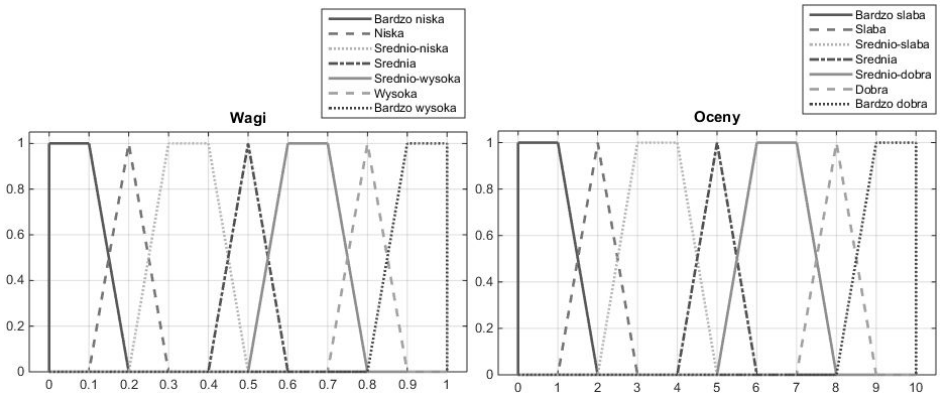
W pierwszej iteracji procedury wielokryterialnej, wspomagającej dobór edytora ontologii, zastosowano metodę wydzielania dla minimalnej wartości atrybutu. Rozpatrywany zbiór kryteriów obejmował następujące charakterystyki edytorów:

- obsługiwane języki ontologii: OWL lub OWL 2,
- obsługa reguł,
- obsługa języka SPARQL,
- wykorzystanie mechanizmów wnioskujących,

- możliwość wizualizacji ontologii,
- dostępność narzędzi do integracji ontologii,
- dostępność narzędzi do wersjonowania ontologii,
- wsparcie pracy grupowej,
- możliwość rozszerzenia funkcjonalności edytora poprzez zewnętrzne rozszerzenia,
- dostępność dokumentacji do edytora,
- wsparcie producenta lub użytkowników.

Na wstępie stwierdzono, że wszystkie edytory obsługują język OWL lub OWL 2. Jest to ważne, ponieważ język OWL jest szeroko stosowany do konstruowania ontologii [Cardoso, 2007] oraz jest on stale wzbogacany i rozszerzany o nowe konstrukcje, a edytory ontologii powinny podążać za jego ewolucją [Casellas, 2011]. Podobnie wszystkie edytory oferują mechanizmy wnioskujące, potrafiące co najmniej sprawdzać ograniczenia i klasyfikować koncepty ontologii. W następnym kroku odrzucono edytory Hozo i Dogma Modeler, które nie umożliwiają stosowania w ontologii języków regułowych (np. RIF bądź SWRL) oraz zapytań SPARQL. Brak tych narzędzi może ograniczać możliwości wnioskowania w ontologii (reguły) i pozyskiwania informacji z ontologii za pomocą zapytań (SPARQL). Wymienione wyżej edytory nie zawierają także wielu innych przydatnych funkcji. Odrzucono również edytor TopBraid Composer w wersji Free Edition, który nie oferuje żadnego narzędzia do wizualizacji ontologii w postaci diagramów graficznych oraz nie jest dla niego dostępne szerokie wsparcie użytkowników lub producenta. Z dalszej analizy wyeliminowano też program KAON2, ze względu na to, że nie zawiera on narzędzi do integracji i wersjonowania ontologii, nie jest dla niego dostępna w zasadzie żadna dokumentacja i wsparcie, a także nie daje możliwości rozszerzania jego funkcjonalności przez zewnętrzne rozszerzenia plug-in. W efekcie przeprowadzenia pierwszej iteracji procedury doboru edytora ontologii do dalszego rozpatrzenia pozostawiono edytory: Protege 5, NeOn Toolkit, Onto Studio oraz Top Braid Composer w wersjach Standard i Maestro. Narzędzia te w różnym stopniu spełniają wszystkie podstawowe wymagania, wymienione w tab. 1.

Druga iteracja procedury doboru edytora wymagała bardziej szczegółowego zbadania poszczególnych edytorów. W oparciu o uszczegółowioną analizę zmodyfikowano też listę kryteriów, wybierając tylko te, które różnicują poszczególne edytory, a także wprowadzając kryteria dodatkowe, np. łatwość obsługi edytora. W tej iteracji zastosowano wielokryterialną metodę Fuzzy TOPSIS, którą zaimplementowano w oprogramowaniu MATLAB. W metodzie tej stosuje się oceny lingwistyczne, odpowiadające zbiorom rozmytym. Funkcje przynależności do poszczególnych zbiorów rozmytych przedstawiono na rys. 2.



Rys. 2. Funkcje przynależności do zbiorów rozmytych (ocen lingwistycznych)

Źródło: Opracowanie własne na podstawie: Chen, Lin, Huang [2006].

Oceny eksperckie (w postaciach: lingwistycznej i funkcji przynależności) dla poszczególnych edytorów oraz rozpatrywane kryteria i przypisane im wagi zawiera tab. 2. W tab. 3 przedstawiono natomiast macierz zawierającą wyznaczone ważone znormalizowane wartości wariantów.

Tabela 2. Wagi kryteriów i oceny eksperckie wariantów w metodzie Fuzzy TOPSIS

Kryterium		**Waga kryterium**	**Warianty**			
			Protege	**NeOn Toolkit**	**Onto Studio**	**TopBraid Composer Standard/ Maestro Ed.**
C1	**Obsługa języków ontologii**	Wysoka	Dobra	Bardzo dobra	Średnio-dobra	Dobra
C2	**Dostępność mechanizmów wnioskujących**	Średnio-wysoka	Bardzo dobra	Średnio-dobra	Średnio-słaba	Średnio-dobra
C3	**Dostępność rozszerzeń plug-in**	Średnia	Bardzo dobra	Dobra	Dobra	Dobra
C4	**Dodatkowe funkcje**	Średnio-niska	Bardzo dobra	Dobra	Dobra	Bardzo dobra
C5	**Łatwość obsługi**	Wysoka	Dobra	Średnio-dobra	Dobra	Dobra
C6	**Jakość dokumentacji**	Wysoka	Dobra	Średnio-dobra	Bardzo dobra	Bardzo dobra
C7	**Jakość wsparcia**	Wysoka	Średnio-dobra	Średnia	Bardzo dobra	Bardzo dobra
C8	**Stabilność oprogramowania**	Bardzo wysoka	Dobra	Dobra	Bardzo dobra	Bardzo dobra
C9	**Aktualność**	Średnia	Bardzo dobra	Słaba	Bardzo dobra	Bardzo dobra
C10	**Koszt oprogramowania**	Bardzo wysoka	Bardzo dobra	Bardzo dobra	Średnio-słaba	Słaba

Źródło: Opracowanie własne.

Tabela 3. Ważona znormalizowana macierz decyzyjna

Kryterium	Warianty			
	Protege	**NeOn Toolkit**	**Onto Studio**	**TopBraid Composer Standard/ Maestro Edition**
C1	[0.49,0.64,0.64,0.81]	[0.56,0.72,0.80,0.90]	[0.35,0.48,0.56,0.72]	[0.49,0.64,0.64,0.81]
C2	[0.40,0.54,0.70,0.80]	[0.25,0.36,0.49,0.64]	[0.10,0.18,0.28,0.40]	[0.25,0.36,0.49,0.64]
C3	[0.32,0.45,0.50,0.60]	[0.28,0.40,0.40,0.54]	[0.28,0.40,0.40,0.54]	[0.28,0.40,0.40,0.54]
C4	[0.16,0.27,0.40,0.50]	[0.14,0.24,0.32,0.45]	[0.14,0.24,0.32,0.45]	[0.16,0.27,0.40,0.50]
C5	[0.544,0.71,0.71,0.90]	[0.39,0.53,0.62,0.80]	[0.54,0.71,0.71,0.90]	[0.54,0.71,0.71,0.90]
C6	[0.49,0.64,0.64,0.81]	[0.35,0.48,0.56,0.72]	[0.56,0.72,0.80,0.90]	[0.56,0.72,0.80,0.90]
C7	[0.35,0.48,0.56,0.72]	[0.28,0.40,0.40,0.54]	[0.56,0.72,0.80,0.90]	[0.56,0.72,0.80,0.90]
C8	[0.56,0.72,0.80,0.90]	[0.56,0.72,0.80,0.90]	[0.64,0.81,1,1]	[0.64,0.81,1,1]
C9	[0.32,0.45,0.50,0.60]	[0.040,0.10,0.10,0.18]	[0.32,0.45,0.50,0.60]	[0.32,0.45,0.50,0.60]
C10	[0.64,0.81,1,1]	[0.64,0.81,1,1]	[0.16,0.27,0.40,0.50]	[0.080,0.18,0.20,0.30]

Źródło: Opracowanie własne.

Na podstawie ważonych znormalizowanych wartości wariantów wyznaczono rozmyte rozwiązanie idealne i anty-idealne, zawarte w tab. 4.

Tabela 4. Rozwiązanie idealne i anty-idealne

Kryterium	A^*	A^-
C1	[0.9,0.9,0.9,0.9]	[0.35,0.35,0.35,0.35]
C2	[0.8,0.8,0.8,0.8]	[0.1,0.1,0.1,0.1]
C3	[0.6,0.6,0.6,0.6]	[0.28,0.28,0.28,0.28]
C4	[0.5,0.5,0.5,0.5]	[0.14,0.14,0.14,0.14]
C5	[0.9,0.9,0.9,0.9]	[0.39,0.39,0.39,0.39]
C6	[0.9,0.9,0.9,0.9]	[0.35,0.35,0.35,0.35]
C7	[0.9,0.9,0.9,0.9]	[0.28,0.28,0.28,0.28]
C8	[1,1,1,1]	[0.56,0.56,0.56,0.56]
C9	[0.6,0.6,0.6,0.6]	[0.04,0.04,0.04,0.04]
C10	[1,1,1,1]	[0.08,0.08,0.08,0.08]

Źródło: Opracowanie własne.

W końcowym etapie procedury Fuzzy TOPSIS uzyskano odległości wariantów od rozwiązania idealnego i anty-idealnego, a na ich podstawie wyznaczono współczynnik odległości i ranking wariantów, przedstawiony w tab. 5.

Tabela 5. Odległości wariantów od rozwiązania idealnego i anty-idealnego oraz ranking wariantów

Wariant	Protege	NeOn Toolkit	Onto Studio	TopBraid Composer Standard/ Maestro Edition
d_i^*	2.4515	3.2798	3.0932	2.9026
d_i^-	3.7003	2.8471	3.0501	3.2265
CC_i	0.6015	0.4647	0.4965	0.5264
Ranking	1	4	3	2

Źródło: Opracowanie własne.

W oparciu o wyniki obliczeń metody Fuzzy TOPSIS stwierdzono, że najwyższą stosowalnością do budowy ontologicznych baz wiedzy charakteryzuje się edytor Protege.

Podsumowanie

W artykule przedstawiono zastosowanie metod wielokryterialnej analizy decyzyjnej do wyboru optymalnego edytora ontologii, pozwalającego konstruować bazy wiedzy. Zastosowano tutaj dwie metody wielokryterialne, tj. metodę wydzielania dla minimalnej wartości atrybutu oraz metodę Fuzzy TOPSIS. W wyniku przeprowadzonej analizy stwierdzono, że najlepszym edytorem spośród rozpatrywanych jest oprogramowanie Protege. Program ten zawiera wszystkie niezbędne funkcje, takie jak: możliwość pracy grupowej, wersjonowanie ontologii, narzędzia do wizualizacji ontologii, liczne mechanizmy wnioskujące, a także wsparcie dla języków OWL2, SWRL i SPARQL. Ponadto charakteryzuje się on ciągłym rozwojem, dobrą dokumentacją i szerokim wsparciem użytkowników, którego skutkiem jest duża dostępność rozszerzeń dla tego edytora, zwiększających jego funkcjonalność. Co istotne, jest to narzędzie rozprowadzane na zasadach bezpłatnej licencji otwartej, dzięki czemu jest ono bardzo szeroko dostępne. To właśnie sposób licencjonowania i zerowy koszt tego oprogramowania stanowi jego główną przewagę nad rozwiązaniami komercyjnymi, którym Protege dorównuje w aspektach funkcjonalności i wsparcia. Dużą popularność tego edytora potwierdzają w pewnym stopniu także wyniki badania przeprowadzonego w 2007 roku (brak nowszych danych) na temat popularności poszczególnych edytorów ontologii. Badanie to wskazuje, że w tym czasie niemal 70% ontologii było budowanych w wykorzystaniem edytora Protege [Cardoso, 2007].

Niniejszy artykuł jest ostatnią publikacją z cyklu prac poruszających zagadnienia doboru języka reprezentacji, metodyki budowy i edytora ontologii. Dalsze badania będą miały aspekt praktyczny polegający na implementacji ontologii

dziedzinowych, w szczególności obejmujących problemy zarządzania. Ontologie takie mogą w dużym stopniu wspomagać decydenta w rozwiązywaniu problemów decyzyjnych, dostarczając mu niezbędnej wiedzy dziedzinowej, pochodzącej od eksperta oraz uzyskanej przez mechanizm wnioskujący [Ziemba i in., 2015; Ziemba i in., 2016].

Literatura

Abraham A. (2003), *Intelligent Systems: Architectures and Perspectives*, "Studies in Fuzziness and Soft Computing", Vol. 113, s. 1-35.

Adamczewski P. (2006), *Zaawansowane systemy informatyczne w zarządzaniu wiedzą w przedsiębiorstwie* [w:] K. Zimniewicz (red.), *Instrumenty zarządzania we współczesnym przedsiębiorstwie. Analiza krytyczna*, Wydawnictwo Akademii Ekonomicznej w Poznaniu, Poznań, s. 17-24.

Alatrish E.S. (2012), *Comparison of Ontology Editors*, "e-RAF Journal on Computing", Vol. 4, s. 23-38.

Błaszczuk A., Brdulak J., Guzik M., Pawluczuk A. (2004), *Zarządzanie wiedzą w polskich przedsiębiorstwach*, Szkoła Główna Handlowa, Warszawa.

Bukowitz W.R., Williams R.L. (1999), *The Knowledge Management Fieldbook*, Financial Times – Prentice Hall, London.

Cardoso J. (2007), *The Semantic Web Vision: Where Are We?* "IEEE Inteligent Systems", Vol. 22, No. 5, s. 84-88.

Casellas N. (2011), *Methodologies, Tools and Languages for Ontology Design* [w:] N. Casellas (ed.), *Legal Ontology Engineering. Methodologies, Modelling Trends, and the Ontology of Professional Judicial Knowledge*, Springer, Heidelberg, s. 57-107.

Chaudhary D., Yadaav P.K., Singh R.K., Mitra S., Ghaziabad S. (2012), *Integrated Knowledge Base: An Approach to Knowledge Extraction*. "International Journal of Computer Applications", Vol. 6, Special Issue on Issues and Challenges in Networking, Intelligence and Computing Technologies, s. 19-26.

Chen C.T. (2000), *Extensions of the TOPSIS for Group Decision-Making under Fuzzy Environment*, "Fuzzy Sets and Systems", Vol. 114, No. 1, s. 1-9.

Chen C.T., Lin C.T., Huang S.F. (2006), *A Fuzzy Approach for Supplier Evaluation and Selection in Supply Chain Management*, "International Journal of Production Economics", Vol. 102, No. 2, s. 289-301.

Chen R.C., Huang Y.H., Bau C.T., Chen S.M. (2012), *A Recommendation System Based on Domain Ontology and SWRL for Anti-diabetic Drugs Selection*, "Expert Systems with Applications", Vol. 39, No. 4, s. 3995-4006.

Dudycz H. (2015), *Kryteria oceny edytorów ontologii*, „Problemy Zarządzania", vol. 13, nr 2, t. 1, s. 78-87.

Escorcio A.L., Cardoso J. (2007), *Editing Tools for Ontology Construction* [w:] J. Cardoso (ed.), *Semantics Web Services: Theory, Tools and Applications*, IGI Global, Hershey, s. 71-95.

Gomez-Perez J.M., Ruiz C. (2010), *Ontological Engineering and the Semantic Web*, "Studies in Computational Intelligence", Vol. 311, s. 191-224.

Gruber T. (1993), *A Translation Approach to Portable Ontology Specifications*, "Knowledge Acquisition", Vol. 5, No. 2, s. 199-220.

Guitouni A., Martel J.M. (1998), *Tentative Guidelines to Help Choosing an Appropriate MCDA Method*, "European Journal of Operational Research", Vol. 109, No. 2, s. 501-521.

Guzman-Arenas A., Cuevas A.D. (2010), *Knowledge Accumulation through Automatic Merging of Ontologies*, "Expert Systems with Applications", Vol. 37, No. 3, s. 1991-2005.

Haghighi P.D., Burstein F., Zaslavsky A., Arbon P. (2013), *Development and Evaluation of Ontology for Intelligent Decision Support in Medical Emergency Management for Mass Gatherings*, "Decision Support Systems", Vol. 54, s. 1192-1204.

Hartley R.T. (1985), *Representation of Procedural Knowledge for Expert Systems*, IEEE 2nd. Conference on AI Applications.

Hwang C.L., Yoon K. (1981), *Multiple Attribute Decision Making. Methods and Applications. A State-of-the-Art Survey*, Springer, Berlin–Heidelberg–New York.

Ivanović M., Budimac Z. (2014), *An Overview of Ontologies and Data Resources in Medical Domains*, "Expert Systems with Applications", Vol. 41, s. 5158-5166.

Kapoor B., Sharma S. (2010), *A Comparative Study Ontology Building Tools for Semantic Web Applications*, "International Journal of Web & Semantic Technology", Vol. 1, No. 3, s. 1-13.

Kotis K., Vouros G. (2010), *Ontological Tools: Requirements, Design Issues and Perspectives* [w:] R. Poli, M. Healy, A. Kameas (eds.), *Theory and Applications of Ontology: Computer Applications*, Springer, Heidelberg, s. 155-173.

Mikuła B., Pietruszka-Ortyl A., Potocki A. (2002), *Zarządzanie przedsiębiorstwem XXI wieku. Wybrane koncepcje i metody*, Difin, Warszawa.

Mishra R.B., Kumar S. (2011), *Semantic Web Reasoners and Languages*, "Artificial Intelligence Review", Vol. 35, No. 4, s. 339-368.

Mizoguchi R., Kozaki K. (2009), *Ontology Engineering Environments* [w:] S. Staab, R. Studer (eds.), *Handbook On Ontologies. Second Edition. International Handbooks on Information Systems*, Springer, Heidelberg, s. 315-336.

Rao A.R., Jain R. (1988), *Knowledge Representation and Control in Computer Vision Systems*, "IEEE Expert", Vol. 3, No. 1, s. 64-79.

Roy B. (1996), *Multicriteria Methodology for Decision Aiding*, Springer, Dordrecht.

Segev A., Gal A. (2008), *Enhancing Portability with Multilingual Ontology-based Knowledge Management*, "Decision Support Systems", Vol. 45, No. 3, s. 567-584.

Valaski J., Malucelli A., Reinehr S. (2012), *Ontologies Application in Organizational Learning: A Literature Review*, "Expert Systems with Applications", Vol. 39, s. 7555-7561.

Wątróbski J., Ziemba P. (2015), *Analiza metodyk budowy ontologii zasobów internetowych*, „Informatyka Ekonomiczna", nr 2(36), s. 60-73.

Ziemba P., Jankowski J., Wątróbski J., Becker J. (2015) *Knowledge Management in Website Quality Evaluation Domain*, "Lecture Notes in Artificial Intelligence", Vol. 9330, s. 75-85.

Ziemba P., Jankowski J., Wolski W. (2015), *Dobór języka reprezentacji wiedzy w ontologiach dziedzinowych*, „Informatyka Ekonomiczna", nr 1(35), s. 84-100.

Ziemba P., Wątróbski J., Jankowski J., Wolski W. (2016), *Construction and Restructuring of the Knowledge Repository of Website Evaluation Methods*, "Lecture Notes in Business Information Processing", Vol. 243, s. 29-52.

[www 1] https://webprotege.stanford.edu (dostęp: 17.03.2017).

[www 2] http://protege.stanford.edu/ (dostęp: 17.03.2017).

[www 3] http://www.hozo.jp/ (dostęp: 17.03.2017).

[www 4] http://neon-toolkit.org/wiki/Main_Page (dostęp: 17.03.2017).

[www 5] http://kaon2.semanticweb.org/ (dostęp: 17.03.2017).

[www 6] http://www.jarrar.info/Dogmamodeler/ (dostęp: 17.03.2017).

[www 7] http://www.semafora-systems.com/en/products/ontostudio/ (dostęp: 17.03.2017).

[www 8] http://www.topquadrant.com/tools/IDE-topbraid-composer-maestro-edition/ (dostęp: 17.03.2017).

SELECTION OF THE ONTOLOGY PROCESSOR FOR ONTOLOGICAL KNOWLEDGE BASE DEVELOPMENT

Summary: Efficient knowledge management allows to avoid of re-seeking solutions to the problems that have already been resolved. Therefore, knowledge should be stored in knowledge bases which are currently often developed in the ontological form. The most important issues, related to the construction of knowledge bases, are: the selection of knowledge representation language, the selection of methodology of the knowledge base construction, and the selection of the ontology processor to support of the development such a structure. The paper presents, based on multi-criteria decision analysis methods, the procedure of the ontology processor selection, based on the characteristics of this type of tools. Selection was conducted with the use of the Conjunctive (Satisficing) and the Fuzzy TOPSIS methods. As a result of the multi-criterial procedure, the processor with optimum level of individual characteristics necessary in this type of software has been indicated.

Keywords: knowledge base, ontology, ontology processor, multi-criteria decision analysis, Fuzzy TOPSIS.