ZESZYTY NAUKOWE UNIWERSYTETU SZCZECIŃSKIEGO

NR 781 EKONOMICZNE PROBLEMY USŁUG NR 106 2013

ADAM CZARNECKI
TOMASZ SITEK
Politechnika Gdańska

ONTOLOGIE VS. REGUŁY – PORÓWNANIE METOD REPREZENTACJI WIEDZY NA PRZYKŁADZIE DZIEDZINY ZARZĄDZANIA USŁUGAMI INFORMATYCZNYMI

Wprowadzenie

Informatyczne narzędzia reprezentacji wiedzy (*Knowledge Representation* – KR) służą przede wszystkim do uzyskiwania na drodze wnioskowania nowej – to jest nie zadeklarowanej wprost – wiedzy w oparciu o wiedzę lub informacje już zgromadzone.

Pojęcie reprezentacji wiedzy kategoryzuje wartości poznawcze i twórcze wszelkich intelektualnych działań oraz zachowań człowieka. Reprezentacja jest związana z realizacją procesów projektowania systemów i wpływa na skuteczność rozwiązań projektowych. Ważne jest również, że stanowi ona jednocześnie źródło i rezultat procesów informatycznych prowadzonych w celu podejmowania decyzji[1]. Wyróżnia się dwa podstawowe typy symbolicznej reprezentacji wiedzy:

– reprezentacja proceduralna – polegająca na określeniu zbioru procedur, działanie których reprezentuje wiedza o dziedzinie;

[1] L. Bolc, W. Borodziewicz, M. Wójcik, *Podstawy przetwarzania informacji niepewnej i niepełnej*, PWN, Warszawa 1991, s. 14.

– reprezentacja deklaratywna – polegająca na określeniu zbioru specyficznych dla rozpatrywanej dziedziny faktów, stwierdzeń, reguł (z ograniczoną informacją dotyczącą sposobu ich wykorzystania).

Zaletą reprezentacji proceduralnej jest duża efektywność opisywania procesów (np. uznanych praw). Reprezentacja deklaratywna jest łatwiejsza w opisie oraz w formalizacji i to ona stanowi treść badań autorów. Wśród metod KR można wskazać dwie stosunkowo popularne: regułowe bazy wiedzy, ontologie.

W tym artykule autorzy stawiają za cel porównanie obu tych metod reprezentacji wiedzy w formie studium przypadku. Przypadkiem tym będzie dziedzina tzw. dobrych praktyk zarządzania usługami informatycznymi (*Information Technology Service Management* – ITSM) o nazwie *Information Technology Infrastructure Library* (ITIL) w wersji 3. W studium zaprezentowane będą wybrane konstrukcje użyte podczas tworzenia ontologii z wyżej wymienionej dziedziny w towarzystwie ich reprezentacji (lub prób reprezentacji) regułowych. Zatem punktem odniesienia są ontologie będące metodą bardziej złożoną, ale dającą w zamian więcej możliwości reprezentacji wiedzy. Od podejścia opartego na regułach będzie się oczekiwać więc, że pozwoli ono na zamodelowanie tej samej wiedzy z użyciem właściwej sobie ekspresywności.

Językiem opisu ontologii użytym w przedstawionych badaniach jest OWL 2 jako implementacja dialektu *SROIQ*(*D*). Reprezentację regułową oparto na notacji języka Prolog.

1. Metody reprezentacji wiedzy

Przetwarzanie wiedzy i budowa mechanizmów realizujących automatycznie wnioskowanie są znane od lat. Pierwsze próby wykorzystania sformalizowanej wiedzy dla celów decyzyjnych podejmowano już w latach 50. i 60. XX wieku. Badania w tej dziedzinie ukierunkowane były wówczas na opracowanie ogólnych zasad inteligentnego rozwiązywania problemów (prace Newella i Simona oraz ich kontynuacje). Stworzony wtedy przez Newella i Simona tzw. GPS (*General Problem Solver*) potrafił rzeczywiście rozwiązywać pewne zadania logiczne dzięki wbudowanemu algorytmowi

poszukującemu drogę w przestrzeni dostępnych stanów[2]. Koncepcja ta, choć wytyczała kierunki działań następcom, okazała się niewystarczająca[3]. Zasadne okazało się podejście, w którym przetwarzanie wiedzy z konkretnej dziedziny wymagało przede wszystkim obszernej wiedzy domenowej. Aparat logiczny niekoniecznie musiał być złożony[4]. Kluczem do efektywnego wykorzystania wiedzy przez mechanizmy wnioskujące stał się więc wybór odpowiedniej metody reprezentacji wiedzy.

Jednymi z najważniejszych metod reprezentacji wiedzy deklaratywnej w systemach wspomagania decyzji są fakty i/lub reguły oraz ontologie, które scharakteryzowano pokrótce poniżej.

1.1. Fakty i reguły

Fakty (lub inaczej stwierdzenia) dotyczą takich zagadnień, jak zdarzenia, zjawiska, objawy, czynności. Stwierdzenia najczęściej zapisywane są w postaci uporządkowanej trójki O-A-V (*Object-Attribute-Value*):

(<OBIEKT> , <ATRYBUT> , <WARTOŚĆ>) (1)

Zwykle zbiór stwierdzeń (faktów) nie jest wystarczający, by opisać dziedzinę w sposób kompletny. Bazy wiedzy, w których oprócz stwierdzeń zawarte są także reguły, stanowią podstawę działania większości systemów ekspertowych powstałych dotychczas. Ogólna postać reguły wygląda następująco:

JEŻELI <przesłanka> TO <konkluzja> (2)

Specjalnie na potrzeby technologii inteligentnych opracowano specyficzne języki programowania, zwane językami symbolicznymi. Z założenia języki

[2] A. Newell, *A guide to the general problem-solver program GPS-2-2*, Rand Corp., Santa Monica, California 1963.

[3] A. Newell, J.C. Shaw, H.A. Simon, *Report on a general problem-solving program*, „Proceedings of the International Conference on Information Processing" 1959, Vol. 2, No. 5222, s. 256–264.

[4] A. Niederliński, *Regułowo-modelowe systemy ekspertowe rmse*, Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, Gliwice 2006, s. 31.

symboliczne miały ułatwić projektowanie systemów inteligentnych, ponieważ oferowały procedury przeznaczone do kodowania wiedzy, konstruowania mechanizmów wnioskujących, przeprowadzania operacji na wyrażeniach symbolicznych w postaci złożonych m.in. z faktów, zdań i reguł[5].

Jednym z najważniejszych języków tej klasy jest Prolog (fr. *Programmation en Logique*). Został stworzony w latach 70. XX wieku, by umożliwić odwzorowanie złożonych relacji logicznych. Prolog pozwolił więc na tzw. programowanie w logice, a dokładniej w języku klauzul Horna (implikacji mających tylko jedną konkluzję). Koncepcja Prologu oparta jest na logice predykatów, dowodzenie twierdzeń zaś polega na rezolucyjnym systemie zaprzeczeń. Na tle klasycznych języków programowania Prolog wyróżnia się możliwością interpretacji kodu zarówno w sposób proceduralny, jak i deklaratywny[6]. Ponieważ linie kodu są bezpośrednio zapisem relacji logicznych, więc mogą być odczytywane jako deklaracje pewnych zależności między zdaniami. Uruchomienie programu napisanego w Prologu powoduje wykonanie procesu dedukcji, czyli wyciąganie wniosków z przesłanek zidentyfikowanych jako dane wejściowe w problemie logicznym. Tym samym zadanie projektanta algorytmu sprowadza się do zdefiniowania tego, „co jest problemem", a nie – jak to jest w przypadku języków proceduralnych – „jak rozwiązać problem"[7]. Istnieje wiele implementacji tego języka różniących się środowiskiem uruchomieniowym, dołączanymi bibliotekami czy chociażby edytorem kodu. Jedną z najbardziej rozpowszechnionych jest darmowy SWI- Prolog[8]. Ta implementacja została też wykorzystana przez autorów na potrzeby prowadzonych badań.

[5] J.J. Mulawka, *Systemy ekspertowe*, Wydawnictwa Naukowo-Techniczne, Warszawa 1997.

[6] T. Sitek, *Ocena języków systemów ekspertowych dla celu implementacji baz wiedzy Systemu Wieloagentowego*, w: *Zarządzanie technologiami informatycznymi: przykłady zastosowań IT*, red. C. Orłowski, Pomorskie Wydawnictwo Naukowo-Techniczne, Gdańsk 2007, s. 102–112.

[7] T. Sitek, *Technologie informatyczne wykorzystywane w projektowaniu i implementacji systemów inteligentnych*, w: *Zarządzanie technologiami informatycznymi: stan i perspektywy rozwoju*, red. C. Orłowski, Pomorskie Wydawnictwo Naukowo-Techniczne, Gdańsk 2006, s. 69–82.

[8] *SWI-Prolog's home*, http://www.swi-prolog.org (5.08.2013).

1.2. Ontologie

Termin „ontologia" informatyka zapożyczyła w drugiej połowie lat 60. XX w. z filozofii – jest to nazwa dyscypliny zajmującej się badaniem istoty bytu. Dość powszechnie pod tym pojęciem rozumie się zbiór ściśle zdefiniowanych pojęć (słownictwo) na temat określonej dziedziny (domeny) akceptowany przez społeczność związaną z ową dziedziną[9]. W definicji nie pojawia się co prawda słowo „model", ale można przyjąć, że tym właśnie dla dziedziny jest ontologia – modelem opisującym pojęcia i ich wzajemne powiązania lub, prościej, modelem reprezentacji wiedzy.

Definicja może nie być wystarczająca, by odpowiedzieć na pytanie, dlaczego warto tworzyć ontologie. Można wskazać co najmniej kilka powodów[10]. Ontologie tworzy się:

- by szerzyć wspólne rozumienie struktury informacji wśród ludzi lub aplikacji agentowych,
- by umożliwić wielokrotne wykorzystanie wiedzy z danej dziedziny,
- by otwarcie sprecyzować założenia odnośnie do wybranej dziedziny,
- by rozdzielić wiedzę o dziedzinie od wiedzy związanej z operowaniem dziedziną,
- by analizować wiedzę o konkretnej dziedzinie.

W praktyce wyróżnia się ontologie o różnym stopniu sformalizowania – od predefiniowanego słownictwa po modele wiedzy oparte na logice[11], zwłaszcza logice opisowej (*Description Logic* – DL). Na tej ostatniej postaci bazuje język OWL (*Web Ontology Language*) w odmianie DL (wersja 1) oraz jego nowsza wersja OWL 2[12]. Jest to język o dużej ekspresywności, co niesie też ze sobą ryzyko stworzenia ontologii, dla której nie uda się w skończonym czasie dokonać poprawnego wnioskowania. Właśnie OWL 2 użyto w badaniach

[9] M. Aufaure, B. Le Grand, M. Soto, N. Bennacer, *Metadata and Ontology-Based Semantic Web Mining*, w: *Web Semantics Ontology*, red. D. Taniar, J.W. Rahayu, Idea Group Publishing, Hershey, London, Melbourne, Singapore 2006, s. 267.

[10] A. Czarnecki, *Technologie informatyczne wykorzystywane w projektowaniu i implementacji ontologii*, w: *Zarządzanie technologiami*…, 2006, s. 84.

[11] K. Goczyła, *Ontologie w systemach informatycznych*, Akademicka Oficyna Wydawnicza „Exit", Warszawa 2011.

[12] P. Hitzler, B. Parsia, P.F. Patel-Schneider, S. Rudolph, *OWL 2 Web Ontology Language Primer (Second Edition)*, http://www.w3.org/TR/owl2-primer/ (5.08.2013).

zaprezentowanych w niniejszej publikacji. O wyborze, oprócz wspomnianej ekspresywności, zdecydowało uznanie tego języka za rekomendowany (a zatem standard de facto) przez World Wide Web Consortium (W3C) dla technologii Semantic Web oraz dostępność narzędzi modelowania ontologii w oparciu o OWL (aplikacja Protégé 4.X).

Należy jeszcze wspomnieć w kontekście niniejszego tekstu, że istnieje rozszerzenie OWL służące do zapisu regułowego — *Semantic Web Rule Language* (SWRL)[13]. Jednak nie zastępuje ono klasycznych dla ontologii wyrażeń, a dodaje możliwość skorzystania z reguł zgodnych z klauzulami Horna. Zatem jego występowanie nie obala głównej myśli niniejszego opracowania – wskazania (na ile to możliwe) regułowych zastępników stwierdzeń ontologicznych.

2. Dziedzina

Do zilustrowania przykładów porównujących zastosowanie konstrukcji ontologicznych i regułowych wybrano wiedzę zawartą w publikacjach opisujących zbiór dobrych praktyk zarządzania usługami informatycznymi (*Information Technology Service Management* – ITSM) zatytułowany *Information Technology Infrastructure Library* (ITIL) w wersji 3. Te publikacje to pięć książek, każda poświęcona jednej z faz składających się na cały cykl:

- strategia świadczenia usług IT (*Service Strategy*)[14],
- projektowanie usług (*Service Design*)[15],
- przekazanie usług (*Service Transition*)[16],
- eksploatacja usług (*Service Operation*)[17],
- ustawiczne doskonalenie usług (*Continual Service Improvement*)[18].

[13] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*, http://www.w3.org/Submission/SWRL/ (5.08.2013).

[14] M. Iqbal, M. Nieves, *Service Strategy*, Office of Government Commerce, London 2007.

[15] V. Lloyd, C. Rudd, *Service Design*, Office of Government Commerce, London 2007.

[16] S. Lacy, I. Macfarlane, *Service Transition*, Office of Government Commerce, London 2007.

[17] D. Cannon, D. Wheeldon, *Service Operation*, Office of Government Commerce, London 2007.

[18] G. Case, G. Spalding, *Continual Service Improvement*, Office of Government Commerce, London 2007.

ITIL wymieniany jest na czele standardów wykorzystywanych przez organizacje wsparcia informatycznego do wsparcia funkcji zarządzania, dając szereg wytycznych na temat świadczenia usług IT, a tym samym osiągania wyższego poziomu dojrzałości i wydajności

Sam zakres pojęciowy użyty w ITIL jest szeroki i jego przedstawienie wykracza poza wąskie ramy tej publikacji. Wybór dziedziny ITIL do demonstracji wyrażeń języka OWL i podejścia regułowego wynika z faktu prowadzenia przez autorów badań w tym obszarze, co znalazło odzwierciedlenie we wcześniejszych publikacjach[19].

3. Koncepty

Koncepty w ujęciu ontologii to formalne reprezentacje pojęć ze świata rzeczywistego. Ich implementacje w języku OWL nazywane są klasami. Klasy te można rozumieć jako zbiory mogące być ogólnymi kategoriami klas bardziej szczegółowych (wówczas tworzy się hierarchia konceptów) lub mieścić w sobie konkretne osobniki.

Aby zaprezentować wiedzę o konceptach oraz zachodzących między nimi związkach, można stosować – często łącznie – wyrażenia logiczne, takie jak subsumpcja, równoważność, rozłączność czy negacja, co zaprezentowano poniżej. Należy jednak zaznaczyć, że poniższe przykłady nie wyczerpują gamy konstrukcji dostępnych inżynierowi wiedzy. Z powodu ograniczonego miejsca nie uwzględniono wyrażeń z użyciem ról i kwantyfikatorów.

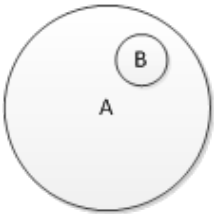
[19] A. Czarnecki, C. Orłowski, *Application of Ontology In the ITIL Domain*, w: *Information Systems Architecture and Technology: Service Oriented Networked Systems*, red. A. Grzech, L. Borzemski, J. Świątek, Z. Wilimowska, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2011, s. 99–108; J. Pastuszak, A. Czarnecki, C. Orłowski, *Ontologically Aided Rule Model for the Implementation of ITIL Processes*, w: *Advances in Knowledge-Based and Intelligent Information and Engineering Systems*, red. M. Grañ, C. Toro, J. Posada, R.J. Howlett, L.C. Jain, IOS Press, 2012, s. 1428–1438; J. Pastuszak, A. Czarnecki, C. Orłowski, *Ontology-Driven Rule-Based Model for an Extension of Information Technology Infrastructure Library Processes*, „Cybernetics and Systems" 2013, Vol. 44, No. 2–3, s. 245–263.

3.1. Hierarchia konceptów

Tworzenie hierarchii pojęć jest jednym z najczęstszych zastosowań ontologii. Opiera się ono na zastosowaniu subsumpcji opisywanej w logice opisowej następująco:

$$B \sqsubseteq A \tag{3}$$

gdzie koncept A subsumuje koncept B (rys. 1).



Rys. 1. Subsumpcja konceptu B przez koncept A
Źródło: opracowanie własne.

Subsumpcja ma charakter przechodni, a zatem zachodzi klasyczny sylogizm:

$$B \sqsubseteq A \wedge C \sqsubseteq B \Rightarrow C \sqsubseteq A \tag{4}$$

Realizacją tej konstrukcji w języku OWL jest wyrażenie:

$$\text{SubClassOf(B A)} \tag{5}$$

Oznacza to, że koncept *B* jest zawarty w koncepcie *A*. Praktycznym zastosowaniem hierarchii konceptów w ontologii ITIL będzie wskazanie, że Service Desk należy do zbioru funkcji ITIL:

$$\text{SubClassOf(Service\_Desk ITIL\_Function)} \tag{6}$$

Natomiast każda funkcja ITIL jest aktywem według tego standardu, co zapisuje się:

$$\text{SubClassOf(ITIL\_Function ITIL\_Asset)} \tag{7}$$

Zgodnie z równaniem (4) z równań (6) i (7) można wywnioskować:

$$\text{SubClassOf(Service\_Desk ITIL\_Asset)} \tag{8}$$

Zaprezentowanie tej samej wiedzy poprzez reguły to konieczność zapisu dwóch faktów dla każdej hierarchii oraz jednej reguły ogólnej:

$$\text{subClassOf(iTILFunction,serviceDesk).} \tag{9}$$

$$\text{subClassOf(iTILAsset,iTILFunction).} \tag{10}$$

$$\text{subClassOf(X,Z) :- subClassOf(X,Y), subClassOf(Y,Z).} \tag{11}$$

Reguła działa w tym przypadku na zasadzie rekurencji i pozwala maszynie wnioskującej języka Prolog pozyskać informacje o wszystkich możliwych zależnościach hierarchicznych, nawet wielokrotnie zagnieżdżonych.

3.2. Równoważność konceptów

O równoważności czy tożsamości konceptów mówimy wtedy, gdy osobniki będące wystąpieniami tych konceptów są zawsze identyczne[20]. Zapisujemy to następująco:

$$B \equiv A \tag{12}$$

Ten sam sens w języku OWL oddaje się za pomocą konstrukcji:

$$\text{EquivalentClasses(B A)} \tag{13}$$

Dla przyjętej w tym tekście dziedziny ITIL następujący zapis:

$$\text{EquivalentClasses(ITIL\_Core\_Book ITIL\_Phase)} \tag{14}$$

[20] K. Goczyła, *op. cit.*, s. 115.

oznacza, że koncept *ITIL_Core_Book*, oznaczający jedną z pięciu książek poświęconych zasadniczym fazom zarządzania usługami informatycznymi, jest równoważny konceptowi *ITIL_Phase*, czyli samej fazie ITIL.

Język Prolog będzie wymagał w takim przypadku przede wszystkim reguły mówiącej o przemienności, jaką charakteryzuje się równoważność:

$$\text{equivalentClasses(X,Y) :- equivalentClasses(Y,X).} \tag{15}$$

Następnie wszystkie tego rodzaju zależności między konceptami będą wymagały określenia ich faktami. Przykładowy fakt będzie miał następującą postać:

$$\text{equivalentClasses(iTIL\_Core\_Book, iTIL\_Phase).} \tag{16}$$

Trzeba przy tym zauważyć, iż założono tu, że tożsamość będzie zawsze relacją między dwoma konceptami. W przypadku większej liczby tożsamych elementów powyższy przykład musiałby być bardziej skomplikowany (dla trzech zmiennych samych reguł wskazujących na przemienność byłoby już sześć).

3.3. Rozłączność konceptów

Rozłączność dwóch konceptów oznacza, że nie może istnieć osobnik, który należałby jednocześnie do obu z nich[21]. Zatem część wspólna obu rozłącznych konceptów jest konceptem pustym ($\bot$), co możemy zapisać:

$$A \sqcap B \sqsubseteq \bot \tag{17}$$

Rozłączność konceptów i w języku OWL zapisuje się następująco:

$$\text{DisjointClasses(A B)} \tag{18}$$

Wymienienie większej niż dwie liczby konceptów rozłącznych należy interpretować jako rozłączność każdej pary konceptów.

Deklarowanie rozłączności konceptów jest jedną z częściej stosowanych konstrukcji przy tworzeniu ontologii opartych na logice opisowej. Wynika to z założenia o świecie otwartym (*Open World Assumption* – OWA), w którym

[21] K. Goczyła, *op. cit.*, s. 116.

dwa koncepty nie są uznawane za różne od siebie tak długo, jak długo nie wyrazi się tego wprost lub nie uda się tego wywnioskować z innych przesłanek.

Przykład zastosowania konstrukcji rozłączności konceptów w ontologii ITIL przedstawiono poniżej:

DisjointClasses (ITIL_Application ITIL_People) (19)

Zapis (19) oznacza, że dwa koncepty będące szczegółowymi wystąpieniami konceptu *ITIL_Asset* (tu niewymienionego), będącego swoistym zbiorem pojęć dotyczących aktywów ITIL, a dotyczące (*ITIL_Application*) aplikacji i ludzi (*ITIL_People*) zaangażowanych w zarządzanie usługami informatycznymi, są rozłączne.

Jeżeli chodzi o zapis regułowy rozłączności, to jest on oparty na tych samych założeniach jak w przypadku tożsamości. Będzie to więc jedna reguła ogólna i odpowiednio tyle faktów, ile wynika z zaobserwowanych relacji rozłączności:

disjointClasses(X,Y) :- disjointClasses (Y,X). (20)

disjointClasses (iTIL_Application,iTIL_People). (21)

3.4. Negacja konceptu

Niekiedy by powiedzieć, czym dana rzecz jest, najłatwiej jest wskazać, czym ona nie jest. To jedno z możliwych zastosowań konstrukcji, jaką jest negacja konceptu. By zapisać, że koncept nie jest konceptem, użyjemy następującej notacji:

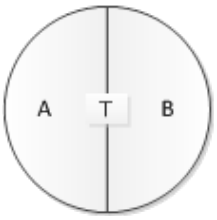
$$B \sqsubseteq \neg A \tag{22}$$

Przenosząc negację na poziom ontologii i języka OWL, powinniśmy patrzeć na koncepty jako na klasy będące swoistymi zbiorami potencjalnych osobników. Wówczas stwierdzenie „nie *A*" oznacza „wszystko, tylko nie *A*". Prawdziwe zatem są następujące tożsamości:

$$\top \equiv A \sqcup B \tag{23}$$

$$\bot \equiv A \sqcap B \tag{24}$$

Oznaczają one, że i – rozłącznie – wypełniają całe uniwersum ⊤. Przedstawiono to na rysunku 2.



Rys. 2. Rozłączność konceptów *A* i *B* wyraża negację i objęcie całego uniwersum ⊤
Źródło: opracowanie własne.

Zapisanie (22) w języku OWL będzie miało następującą postać:

ClassAssertion(ObjectComplementOf (A) B) (25)

Nawiązując do ontologii ITIL, można chcieć przedstawić wiedzę o tym, że organizacja nie stosująca tego zbioru dobrych praktyk (*NonITILOrganization*) jest przeciwieństwem organizacji, w której elementy ITIL wdrożono (*ITILOrganization*). Należy przy tym zawęzić obszar pojęciowy tylko do konceptów subsumowanych przez pojęcie organizacji, co można osiągnąć przez zastosowanie przecięcia konceptów (części wspólnej). Zatem opisana tu wiedza może zostać wyrażona tak:

EquivalentClasses(NonITILOrganization ObjectIntersectionOf
(Organization ObjectComplementOf(ITILOrganization))) (26)

W jaki sposób należałoby to zrobić w deklaratywnej bazie wiedzy języka Prolog? Istotę negacji trzeba wyrazić regułą z zastosowaniem predykatu „not".

organizationType(iTILOrganization, organization(X)) :-
not(organizationType(nonITILOrganization, organization(X))). (27)

By skorzystać z tej reguły, trzeba byłoby każdą nowo dodaną organizację definiować również z punktu widzenia przynależności do grupy podmiotów

posiadających ITIL lub nie. Przykładowe dwa poniższe fakty pozwalają wysnuć wniosek o tym, że Politechnika Gdańska nie zalicza się do tego grona.

organization(politechnikaGdanska) (28)

organizationType
(nonITILOrganization,organization(politechnikaGdanska)). (29)

4. Zestawienie form reprezentacji wiedzy

Wykorzystane w niniejszym tekście konstrukcje zestawiono w tabeli 1. Jak widać, udało się znaleźć regułowe odpowiedniki form ontologicznych. Niemniej, z wyjątkiem negacji, każdy z przykładów regułowych wymaga większej liczby deklaracji, by udało się wyrazić tę samą wiedzę o konceptach.

Dodatkowo, czego nie udało się uwzględnić w niniejsze publikacji z uwagi na ograniczoną objętość artykułu, autorzy zbadali konstrukcje z użyciem ról oraz kwantyfikatorów: szczegółowego i ogólnego. Dla ich reprezentacji ontologicznej nie udało się znaleźć satysfakcjonującej postaci regułowej.

Stąd wydaje się zasadne stwierdzenie, że dla użytej w badaniach dziedziny ITIL, opartej przede wszystkim na relacjach zachodzących między pojęciami, podejście wykorzystujące ontologie daje większe możliwości zamodelowania dziedziny przy mniejszej złożoności stosowanych konstrukcji wybranego języka.

Tabela 1

Zestawienie ontologicznych i regułowych reprezentacji wiedzy

Typ wiedzy	**DL**	**Ontologia (OWL)**	**Reguły (Prolog)**
Hierarchia konceptów	$C \sqsubseteq B \sqsubseteq A$	SubClassOf(*B A*) SubClassOf(*C B*)	subClassOf(X,Z) :- subClassOf(X,Y), subClassOf(Y,Z)
Równoważność konceptów	$B \equiv A$	EquivalentClasses(*B A*)	equivalentClasses(X,Y) :- equivalentClasses(Y,X)
Rozłączność konceptów	$B \sqcap A \sqsubseteq \bot$	DisjointClasses(*B A*)	disjointClasses(X,Y) :- disjointClasses (Y,X)
Negacja konceptów	$B \sqsubseteq \neg A$	ClassAssertion (ObjectComplementOf(*A*) *B*)	classAssertion(Y, X) :- not(classAssertion(Z, X))

Źródło: opracowanie własne.

Podsumowanie

W tekście zaprezentowano przegląd i zgrubne porównanie ontologicznych i regułowych metod reprezentacji wiedzy dla wybranych typowych zagadnień modelowania pojęć i ich zależności. Celem tego porównania było poszukiwanie odpowiedzi na pytanie o słuszność stosowania ontologii, a nie reguł dla obranej dziedziny, jaką jest zbiór dobrych praktyk ITIL.

Prezentacja wyników została ograniczona jedynie do podstawowych wyrażeń opisujących koncepty w dziedzinie. Niemniej nawet na takim poziomie ogólności uwidoczniła się przewaga podejścia opartego na ontologiach i języku OWL nad parą reguły–Prolog. Wynika ona z bogatszego podłoża semantycznego kryjącego się za elementami języka OWL, co wynika z oparcia go na co najmniej dwóch niższych warstwach językowych – *Resource Description Framework* (RDF) i RDF Schema, będących odpowiednio: metametajęzykiem i metajęzykiem dla OWL. W takim ujęciu Prolog jest raczej na poziomie metametajęzyka.

Do pełniejszego obrazu porównania zabrakło innych ważnych konstrukcji, choćby ról łączących koncepty i stosowanych razem z nimi ograniczeń, takich jak „istnieje", „wszystkie", „co najmniej", „co najwyżej", „dokładnie". Poza tym w przedstawionych przykładach udało się zilustrować pewne hipotezy mówiące o większej przydatności reprezentacji ontologicznej, natomiast nie przeprowadzono tu żadnego formalnego dowodu. Te zagadnienia będą uwzględnione przez autorów w dalszych badaniach i publikacjach.

Literatura

Aufaure M., Le Grand B., Soto M., Bennacer N., *Metadata and Ontology-Based Semantic Web Mining*, w: *Web Semantics Ontology*, eds. D. Taniar, J.W. Rahayu, Idea Group Publishing, Hershey, London, Melbourne, Singapore 2006.

Bolc L., Borodziewicz W., Wójcik M., *Podstawy przetwarzania informacji niepewnej i niepełnej*, PWN, Warszawa 1991.

Cannon D., Wheeldon D., *Service Operation*, Office of Government Commerce, London 2007.

Case G., Spalding G., *Continual Service Improvement*, Office of Government Commerce, London 2007.

Czarnecki A., *Technologie informatyczne wykorzystywane w projektowaniu i implementacji ontologii*, w: *Zarządzanie technologiami informatycznymi: stan i perspektywy rozwoju*, red. C. Orłowski, Pomorskie Wydawnictwo Naukowo-Techniczne, Gdańsk 2006.

Czarnecki A., Orłowski C., *Application of Ontology In the ITIL Domain*, w: *Information Systems Architecture and Technology: Service Oriented Networked Systems*, red. A. Grzech, L. Borzemski, J. Świątek, Z. Wilimowska, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2011.

Goczyła K., *Ontologie w systemach informatycznych*, Akademicka Oficyna Wydawnicza EXIT, Warszawa 2011.

Hitzler P., Parsia B., Patel-Schneider P.F., Rudolph S., *OWL 2 Web Ontology Language Primer,* second edition, http://www.w3.org/TR/owl2-primer/.

Horrocks I., Patel-Schneider P.F., Boley H., Tabet S., Grosof B., Dean M., *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*, http://www.w3.org/Submission/SWRL/.

Iqbal M., Nieves M., *Service Strategy*, Office of Government Commerce, London 2007.

Lacy S., Macfarlane I., *Service Transition*, Office of Government Commerce, London 2007.

Lloyd V., Rudd C., *Service Design*, Office of Government Commerce, London 2007.

Mulawka J.J., *Systemy ekspertowe*, Wydawnictwa Naukowo-Techniczne, Warszawa 1997.

Newell A., *A guide to the general problem-solver program GPS-2-2*, Rand Corp., Santa Monica, California 1963.

Newell A., Shaw J.C., Simon H.A., *Report on a general problem-solving program*, „Proceedings of the International Conference on Information Processing" 1959, t. 2, nr 5222.

Niederliński A., *Regułowo-modelowe systemy ekspertowe rmse*, Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, Gliwice 2006.

Pastuszak J., Czarnecki A., Orłowski C., *Ontologically Aided Rule Model for the Implementation of ITIL Processes*, w: *Advances in Knowledge-Based and Intelligent Information and Engineering Systems*, eds. M. Graña., C. Toro, J. Posada, R.J. Howlett, L.C. Jain, IOS Press 2012.

Pastuszak J., Czarnecki A., Orłowski C., *Ontology-Driven Rule-Based Model for an Extension of Information Technology Infrastructure Library Processes*, „Cybernetics and Systems" 2013, t. 44, nr 2–3.

Sitek T., *Ocena języków systemów ekspertowych dla celu implementacji baz wiedzy Systemu Wieloagentowego*, w: *Zarządzanie technologiami informatycznymi: przykłady zastosowań IT*, red. C. Orłowski, Pomorskie Wydawnictwo Naukowo-Techniczne, Gdańsk 2007.

Sitek T., *Technologie informatyczne wykorzystywane w projektowaniu i implementacji systemów inteligentnych*, w: *Zarządzanie technologiami informatycznymi: stan i perspektywy rozwoju*, red. C. Orłowski, Pomorskie Wydawnictwo Naukowo-Techniczne, Gdańsk 2006.

SWI-Prolog's home, http://www.swi-prolog.org.

ONTOLOGIES VS. RULES – COMPARISON OF KNOWLEDGE REPRESENTATION METHODS BASED ON THE IT SERVICES MANAGEMENT DOMAIN EXAMPLE

Summary

This text provides a brief overview of selected structures aimed at knowledge representation in the form of ontologies based on description logic and aims at comparing them with their counterparts based on the rule-based approach. Due to the limitations on the length of the article, only elements associated with the representation of concepts could be shown, without including roles. The formalisms of the OWL language were used to record ontologies, while the rules were expressed in Prolog.

To better illustrate these two ways of knowledge representation, examples of best practices from the field of IT services management were used, which are contained in a set of publications known as the Information Technology Infrastructure Library (ITIL).

The purpose of the comparison is to examine the possibility of using an ontological approach in situations where the use of rule-based solutions is problematic.

Translated by Adam Czarnecki and Tomasz Sitek