ontologie, XML, RDF, OWL, RDFS, Dublin Core,
PROTEGE, OIL, semantyczny Web, Swoogle

WIESŁAW GLIŃSKI
Instytut Informacji Naukowej i Studiów Bibliologicznych, UW

JĘZYKI I NARZĘDZIA DO TWORZENIA I WYSZUKIWANIA ONTOLOGII W KONTEKŚCIE SEMANTYCZNEGO WEBA

Przedstawiono wybrane języki (RDF, RDFS, OWL) do opisu ontologii, ilustrując ich zastosowanie przykładami. Omówiono dwa najbardziej popularne edytory do tworzenia ontologii Protege i OIL oraz wyszukiwarkę dokumentów Semantycznego Weba – SWOOGLE. Całość zakończono rozważaniami na temat rosnącej popularności zastosowania ontologii w Polsce.

1. WSTĘP

Jak pokazano w poprzednim artykule, pojęciu ontologii jako dziedziny inżynierii wiedzy poświęca się coraz więcej uwagi. Choć samo zagadnienie jako część inżynierii wiedzy nie jest nowe, sięga bowiem początku lat 90-tych, to obecnie w kontekście tzw. Semantycznego Weba nabiera dodatkowego znaczenia, można powiedzieć „przeżywa drugą młodość". Będąc pod wrażeniem wizjonerskiego podejścia prezentowanego przez T. Berners-Lee, J. Hendlera i O. Lassila (2001) ontologie wydają się istotnym, jeśli nie kluczowym składnikiem przyszłej sieci WWW pełniąc w niej rolę umożliwiającą sprawne wyszukiwanie dokumentów.

2. JĘZYKI I NARZĘDZIA DO OPISU ONTOLOGII

Tworzenie ontologii wymaga języków, które dzięki sile swej ekspresji umożliwiłyby opisanie ontologii. Obecnie większość znanych języków do opisu ontologii wykorzystuje XML, który jest metajęzykiem. Dzięki niemu możliwe jest stworzenie wielu innych języków pozwalających na opisywanie ontologii. Istotnym elementem XML są tzw. przestrzenie nazw (ang. *namespaces*), które pozwalają na uniknięcie problemów wynikłych z zastosowania tych

samych nazw w znacznikach. Przypomnijmy, że atrybut ten jest wprowadzany na początku każdego znacznika i ma następującą postać:

```
xmlns:namespace-prefix="przestrzeń_nazw"
```

np.:

```
xmlns:f="http://www.w3schools.com/furniture"
```

Kiedy definiujemy przestrzeń nazw w znaczniku początkowym, wszystkie elementy potomne posiadające taki sam prefiks uznaje się za połączone z daną przestrzenią nazw.

2.1. RDF

Język RDF (Resource Description Framework) jest językiem zbudowanym na bazie XML. Pozwala na opisywanie zasobów w sieci Web. Celem RDF jest przedstawianie wiedzy zawartej w tych zasobach w postaci łatwo przetwarzanej przez programy komputerowe, nie zaś ich wyświetlanie użytkownikom.

Przykładem zastosowań RDF może być:

- opisywanie własności zakupywanych produktów np. dostępność, cena itd.;
- opisywanie dat harmonogramu różnego typu wydarzeń w sieci;
- przedstawianie informacji o stronach Webowych (metainformacji), np. data;
- utworzenia dokumentu lub jego modyfikacji, tytuł, autor itd.;
- opisywanie zawartości znaczenia obrazów;
- opisywanie zawartości systemów wyszukiwawczych;
- opisywanie bibliotek cyfrowych.

W celu lokalizowania i opisania zasobów sieci WWW oraz własności RDF wykorzystuje URI (Uniform Resource Identifier). Najbardziej rozpowszechnionym rodzajem URI jest tzw. URL (Uniform Resource Locator), dzięki któremu możliwe jest identyfikowanie adresów nazw domenowych w sieci Internet. Innym choć nie tak popularnym rodzajem URI jest np. URN (Universal Resource Name). Poniższy przykład został celowo uproszczony (pominięto w nim przestrzenie nazw).

```
<?xml version="1.0"?>
<RDF>
<Description about="http://www.lis.uw.edu/default.asp">
<autor>Jan Abacki</autor>
<utworzono>1 Maj 1999</utworzono>
<zmodyfikowano>1 Luty 2004</zmodyfikowano>
</Opis>
</RDF>
```

Przedstawiony powyżej URI :"http://www.lis.uw.edu/default.asp" jest wykorzystywany do identyfikowania zasobów sieci WWW. Własność *autor* opisuje autora strony i przyjmuje wartość „Jan Abacki”. Własność *utworzono* określa, kiedy strona była stworzona, a własność *zmodyfikowano* wskazuje datę jej modyfikacji.

Można powiedzieć, że język RDF operuje identyfikatorami zasobów, własności oraz wartości i tak w naszym przykładzie:

- http://www.lis.uw.edu/default.asp jest zasobem,
- element <autor> jest własnością,
- zaś “Jan Abacki” jest wartością.

Wiedzę w RDF przedstawia się w postaci trójki: podmiot, orzeczenie, dopełnienie (lub: obiekt, rodzaj powiązania, wartość cechy). Tak więc w naszym przykładzie

- http://www.lis.uw.edu/default.asp jest podmiotem (obiektem),
- element <autor> jest orzeczeniem (rodzajem powiązania, predykatem),
- zaś wartość “Jan Abacki” jest dopełnieniem (wartością cechy).

RDF (określane jako RDF/XML) stało się standardem preferowanym przez W3C (World Wide Web Consortium) w lutym 2004 r. Dzięki wykorzystaniu języka XML może być łatwo przetwarzalne przez różne systemy komputerowe, niezależnie od systemu operacyjnego czy aplikacji. Załóżmy, że mamy do czynienia z bazą książek:

Tytuł	Autor	Miejsce Wydania	Wydawnictwo	Cena	Rok Wydania
Basnie	Andersen H. C.	Warszawa	PIW	78	1969
Życie Pi	Martel Y.	Warszawa	Znak	29	2004
…	…	…	…	…	…

Przyjrzyjmy się zatem przykładowemu zbiorowi RDF, który zostanie utworzony na podstawie powyższej tabeli:

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ks="http://www.bookstore.zmyslony/ksiazki">
<rdf:Description
rdf:about="http://www.ksiegarnia.zmyslona/ksiazki/Basnie">
  <ks:autor>Andersen H.C.</ks:autor>
  <ks:miejsceWydania>Warszawa</ks:miejsceWydania>
  <ks:wydawnictwo>PIW</ks:wydawnictwo>
  <ks:cena>78</ks:cena>
  <ks:rok>1969</ks:rok>
```

```
</rdf:Description>
<rdf:Description
rdf:about="http://www.ksiegarnia.zmyslona/ksiazki/Zycie Pi">
  <ks:autor>Martel Y.</ks:autor>
  <ks:miejsceWydania>Warszawa</ks:miejsceWydania>
  <ks:wydawnictwo>Znak</ks:wydawnictwo>
  <ks:cena>29</ks:cena>
  <ks:rok>2004</ks:rok>
</rdf:Description>
.
.
.
</rdf:RDF>
```

Pierwszy wiersz w zbiorze XML, to deklaracja typu dokumentu, określająca wersję XML.

Element rdf:RDF (zaczynający się od rdf:RDF i kończący się na /rdf:RDF) wskazuje, że zawartość jest typu RDF.

Element xmlns:rdf – przestrzeń nazw ustala, że znaczniki z ciągiem znaków rdf: pochodzą z przestrzeni nazw zdefiniowanej przez "http://www.w3.org/1999/02/22-rdf-syntax-ns#".

Przestrzeń nazw xmlns:ks oznacza, że znaczniki z ciągiem znaków ks: należą do przestrzeni zdefiniowanej w http://www.ksiegarnia.zmyslona/ksiazki/.

Element rdf:opis (zaczynający się od rdf:Opis i kończący się na /rdf:Opis) zawiera opis zasobu identyfikowanego przez atrybut rdf:about.

Element rdf:autor opisuje własność zasobu, podobnie jak atrybut ks:miejsceWydania.

Kiedy zastanawiamy się nad RDF, winniśmy mieć na względzie następującą regułę: zasoby mają własności które przyjmują określone wartości". I tak, na podstawie podanego przykładu możemy stwierdzić, że:

– zasób http://www.ksiegarnia.zmyslona/ksiazki/Basnie ma własność zwaną *autor*, która przyjmuje wartość „Andersen H.C.";

– Zasób http://www.ksiegarnia.zmyslona/ksiazki/Basnie ma własność zwaną *cena*, która przyjmuje wartość „78". Oczywiście użytkownik sformułowałby to w następujący sposób: "Baśnie H.C. Andersena kosztują 78 zł".

Przyjrzyjmy się teraz najważniejszym elementom RDF.

Element RDF jest elementem głównym dokumentu RDF. Definiuje dokument XML jako dokument RDF i zawiera odwołanie do przestrzeni nazw xmlns:rdf np.:

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
.
```

```
. Dalej następuje szczegółowy opis.
.
</rdf:RDF>
```

Element główny RDF musi zawsze być zapisany jako <rdf:RDF> oraz mieć odwołanie do przestrzeni nazw http://www.w3.org/1999/02/22-rdf-syntax-ns#.

Kolejnym elementem jest *Description*, który opisuje zasób. Atrybut *about* identyfikuje zasób:

```
   <?xml version="1.0"?>
      <rdf:RDF
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:ks="http://www.bookstore.zmyslony/ksiazki">
     <rdf:Description
     rdf:about="http://www.ksiegarnia.zmyslona/ksiazki/Basnie">
       <ks:autor>Andersen H.C.</ks:autor>
       <ks:miejsceWydania>Warszawa</ks:miejsceWydania>
       <ks:wydawnictwo>PIW</ks:wydawnictwo>
       <ks:cena>78</ks:cena>
       <ks:rok>1969</ks:rok>
     </rdf:Description>
     </rdf:RDF>
```

Elementy dotyczące własności (*autor*, *wydawca*, *miejsceWydania*, *cena*, *rok*) są definiowane w przestrzeni xmln:ks. Ta przestrzeń jest poza RDF (nie jest częścią RDF). RDF definiuje tylko szablon. Własności winny być zdefiniowane przez kogoś innego (przedsiębiorstwo, organizację, osobę, itd.)

Własności mogą być też wyrażone w postaci atrybutów (zamiast elementów):

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ks="http://www.bookstore.zmyslony/ksiazki">
<rdf:Description
rdf:about="http://www.ksiegarnia.zmyslona/ksiazki/Basnie">
ks:autor="Andersen H.C."  ks:miejsceWydania="Warszawa"
ks:wydawnictwo="PIW"  ks:cena="78"
ks:rok="1969"
</rdf:Description>
</rdf:RDF>
```

W powyższym przykładzie własności (*autor*, *miejsceWydania*, *wydawnictwo*, *cena*, *rok*) są wyrażone jako atrybuty, zamiast jako elementy zagnieżdżone wewnątrz znaczników.

Ponadto własności mogą być wyrażone również jako zasoby:

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ks="http://www.bookstore.zmyslony/ksiazki">
<rdf:Description
rdf:about="http://www.ksiegarnia.zmyslona/ksiazki/Basnie"
```

```
  <ks:autor
  rdf:about="http://www.ksiegarnia.zmyslona/autorzy/andersen"/>
  <ks:miejsceWydania>Warszawa</ks:miejsceWydania>
  <ks:wydawnictwo>PIW</ks:wydawnictwo>
  <ks:cena>78</ks:cena>
  <ks:rok>1969</ks:rok>
</rdf:Description>
</rdf:RDF>
```

W podanym przykładzie własność *autor* nie posiada wartości, ale odwołanie do zasobu, w którym jest informacja na temat danego autora.

Kolejnym elementem jest znacznik <rdf:Bag>, określający nieuporządkowany zbiór określonych elementów.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ks="http://www.bookstore.zmyslony/ksiazki">
<rdf:Description
rdf:about="http://www.ksiegarnia.zmyslona/ksiazki/Basnie">
<ks:rok>
 <rdf:Bag>
      <rdf:li>1998</rdf:li>
      <rdf:li>1971</rdf:li>
      <rdf:li>1982</rdf:li>
      <rdf:li>1969</rdf:li>
      <rdf:li>1978</rdf:li>
  </rdf:Bag>
</ks:rok>
</rdf:Description>
</rdf:RDF>
```

W podanym przykładzie własność <ks:rok> zawiera zbiór wartości dla *roku wydania* (czyli identyfikuje poszczególne wydania książki).

Następnym elementem jest tzw. <rdf:Seq>, określający uporządkowany zbiór elementów.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ks="http://www.bookstore.zmyslony/ksiazki">
<rdf:Description
rdf:about="http://www.ksiegarnia.zmyslona/ksiazki/Basnie">
<ks:rok>
 <rdf:Seq>
      <rdf:li>1969</rdf:li>
      <rdf:li>1971</rdf:li>
      <rdf:li>1978</rdf:li>
      <rdf:li>1982</rdf:li>
      <rdf:li>1982</rdf:li>
  </rdf:Seq>
</ks:rok>
</rdf:Description>
</rdf:RDF>
```

W podanym wyżej przykładzie własność <ks:rok> zawiera zbiór wartości dla *roku wydania*, czyli poszczególnych lat wydania ułożonych w porządku rosnącym.

Następnym elementem jest tzw <rdf:Alt>, określający zbiór elementów alternatywnych.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ks="http://www.bookstore.zmyslony/ksiazki">
<rdf:Description
rdf:about="http://www.ksiegarnia.zmyslona/ksiazki/Basnie">
<ks:opr>
 <rdf:Alt>
      <rdf:li>miękka</rdf:li>
      <rdf:li>twarda</rdf:li>
</rdf:Description>
</rdf:RDF>
```

W podanym wyżej przykładzie własność <ks:opr> (oprawa) zawiera zbiór alternatywnych wartości dla atrybutu *oprawa*.

2.2. RDFS I DUBLIN CORE

RDF opisuje zasoby z uwzględnieniem klas, własności oraz wartości. Ponadto RDF potrzebuje sposobu na zdefiniowanie klas specyficznych dla danych aplikacji. W tym celu wykorzystuje się rozszerzenie języka RDF. Jednym z takich rozszerzeń jest tzw. schemat RDF (RDF Schema, RDFS). Schemat RDF dostarcza sposobu pozwalającego opisać aplikacje specyficzne dla danej klasy i własności. Klasy w schemacie RDF są podobne do klas w projektowaniu obiektowym. Dzięki nim możliwe jest zdefiniowanie zasobów jako wystąpień klas i podklas (klas podrzędnych).

Poniższy przykład przedstawia niektóre z ułatwień RDFS:

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf= "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base=  "http://www.zwierzeta.zmyslony/zwierzeta">

<rdf:Description rdf:ID="zwierze">
<rdf:type
  rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

<rdf:Description rdf:ID="kot">
<rdf:type
```

```
  rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
<rdfs:subClassOf rdf:resource="#zwierze"/>
</rdf:Description>

<rdf:Description rdf:ID="pies">
<rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
<rdfs:subClassOf rdf:resource="#zwierze"/>
</rdf:Description>
</rdf:RDF>
```

W podanym przykładzie zasoby „pies", „kot" są podklasą klasy „zwierze".

Jeśli klasa jest zasobem RDF, możemy skrócić zapis używając rdf:Class zamiast rdf:Description i pominąć część rdf:type. Przedstawiony powyżej przykład przekształcimy zatem na:

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf= "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base=  "http://www.zwierzeta.zmyslony/zwierzeta">

<rdfs:Class rdf:ID="zwierze">

<rdfs:Class rdf:ID="kot">
<rdfs:subClassOf rdf:resource="#zwierze"/>
</rdfs:Class>

<rdfs:Class rdf:ID="pies">
<rdfs:subClassOf rdf:resource="#zwierze"/>
</rdfs:Class>

</rdf:RDF>
```

Ciekawym rozwiązaniem jest schemat metadanych Dublin Core, który dostarcza własności opisujących obiekty w sieci, dzięki którym są one lepiej identyfikowalne przez systemy wyszukiwawcze (zob. tab.1). Więcej na ten temat można znaleźć w pracy M. Nahotki (2004).

Przypomnijmy, że RDF jest językiem zapisu metadanych (danych o danych). Własności obiektów proponowane przez Dublin Core zostały zdefiniowane na warsztatach poświęconych metadanym w 1995 r. w Dublinie (Ohio, Stany Zjednoczone) i obecnie są rozwijane przez Grupę Roboczą do prac nad Metadanymi (Dublin Core Metadata Initiative; http://dublincore.org/).

Tabela 1 Podstawowe elementy Dublin Core.

Własność	Definicja
Contributor	Współtwórca
Coverage	Zasięg (zakres) danego zasobu
Creator	Twórca
Format	Format
Date	Data
Description	Opis
Identifier	Identyfikator
Language	Język
Publisher	Wydawca
Relation	Relacja
Rights	Własność (informacje na temat praw własności)
Source	Źródło
Subject	Opis rzeczowy
Title	Tytuł
Type	Typ

Przyjrzyjmy się przykładowemu Schematowi RDF według Dublin Core:

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc= "http://purl.org/dc/elements/1.1/">
<rdf:Description rdf:about="http://www.biblioteka.fikcijna">
<dc:title>Projekt badawczy bibliotek naukowych</dc:title>
<dc:description>
To jest przykładowy program badawczy dotyczący bibliotek publicznych
</dc:description>
<dc:publisher>Jan Abacki</dc:publisher>
<dc:date> 1999-09-01 </dc:date>
<dc:type>World Wide Web Home Page</dc:type>
<dc:format>text/html</dc:format>
<dc:language>pl</dc:language>
</rdf:Description>

</rdf:RDF>
```

2.3. OWL

OWL (Web Ontology Language) jest językiem umożliwiającym przetwarzanie informacji w sieci WWW, zbudowanym jako „rozszerzenie” języka RDF i podobnie jak RDF może być interpretowany przez programy komputerowe. OWL jest częścią wizji Semantycznego Weba, w której informacja ma określone znaczenie, może być przetwarzana przez komputery, zaś komputery mogą łączyć ze sobą i „rozumieć” różnego typu informacje. W zasadzie cele OWL i RDF są zbieżne, lecz OWL posiada większe możliwości interpretacyjne niż RDF, wyposażony jest też w bogatszy słownik oraz składnię.

Zdaniem J. Srzednickiego (2004) ontologia OWL może zawierać następujące elementy:

- usystematyzowaną relację pomiędzy klasami;
- właściwości typu danych, opisujące atrybuty elementów klas;
- właściwości obiektów, opisujące relacje pomiędzy elementami klas;
- instancje klas;
- instancje właściwości.

Ze zbioru twierdzeń zapisanych za pomocą OWL tworzymy bazę wiedzy. Twierdzenia mogą dotyczyć faktów o poszczególnych wystąpieniach klas, ale także różnych faktów pochodnych, wywnioskowanych za pomocą semantyki OWL. Ontologie zwykle nie zawierają instancji klas (w przeciwieństwie do baz wiedzy), często jednak zdarza się, że wystąpienia są potrzebne w ontologii do definiowania konkretnych klas. Przykładowo, gdy weźmiemy *Wino* i *Kolor* jako klasy, i *czerwony* jako wystąpienie klasy *Kolor*, wtedy do definicji klasy *Czerwone wino* potrzebne nam jest owe wystąpienie.

Język OWL występuje w trzech odmianach:

- OWL Lite (zaprojektowany z myślą stworzenia prostego języka, wystarczającego dla użytkowników potrzebujących prostych klasyfikacji i prostych ustalonych cech, np. gdy mamy wprowadzone ograniczenia liczebności cech (ang. *cardinality*), ale mogą one przyjmować wartości jedynie 0 i 1);
- OWL DL (zawiera OWL Lite) zawiera pełny zbiór słownictwa, interpretowanego pod kątem zbioru prostych ustalonych cech;
- OWL Full (zawiera OWL DL) zawiera pełne słownictwo OWL, interpretowane znacznie szerzej niż w wersji OWL DL, z pełnymi możliwościami zapewnianymi przez RDF. Klasa może być tu traktowana równocześnie jako zbiór jednostek i jako jednostka jako taka.

Założeniem semantycznego Weba jest możliwość wykorzystywania ontologii w sposób rozproszony przy założeniu, że nowe informacje nie mogą negować już istniejących. OWL pozwala na negację informacji już istniejących, co zdaniem J. Srzednickiego (2004) pozwala na opisywanie przeciwieństw („mogą się pojawiać nawet w sposób nie jawny”).

Zanim rozpoczniemy tworzenie ontologii wskazane jest stworzenie przestrzeni nazw. Posłużmy się przykładową przestrzenią nazw dla ontologii win [przykład za OWL (2004a)]:

```
<rdf:RDF
    xmlns     ="http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
    xmlns:vin ="http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
    xml:base  ="http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
    xmlns:food="http://www.w3.org/TR/2004/REC-owl-guide-20040210/food#"
    xmlns:owl ="http://www.w3.org/2002/07/owl#"
    xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd ="http://www.w3.org/2001/XMLSchema#">
```

Pierwsze dwie deklaracje dotyczą przestrzeni nazw związanej z konkretną ontologią. Trzecia deklaracja określa podstawowy URI dla dokumentu. Czwarta pełni charakter pomocniczy dla ontologii „produkty spożywcze” (ang. *food*). Po określeniu przestrzeni nazw możemy stworzyć przykładową ontologię [przykład za J. Srzednickim (2004)]:

```
<owl:Ontology rdf:about="http://www.example.org/wine">
  <rdfs:comment>Przykładowa ontologia</rdfs:comment>
  <owl:versionInfo>
      $Id: semantic_web.xml,v 1.4 2003/12/30 00:21:51 winfried Exp $
  </owl:versionInfo>
  <owl:imports rdf:resource="http://www.w3.org/TR/2002/WD-owl-guide-
20021104/food.owl"/>
  </owl:Ontology>
</rdf:RDF>
```

W OWL, każda jednostka jest członkiem klasy <owl:Thing>. Gdy tworzymy ontologię dla danej dziedziny, najpierw tworzona jest klasa główna dla tej dziedziny i dopiero na niej budowana jest hierarchia podklas. W przykładowej ontologii win według J. Strzednickiego (2004) zdefiniowane są trzy główne klasy:

```
<owl:Class rdf:ID="Wino"/>
<owl:Class rdf:ID="Region"/>
<owl:Class rdf:ID="ArtykułSpożywczy"/>
```

Następnie utworzona jest podklasa ArtykułuSpożywczego, o nazwie Napój.

```
<owl:Class rdf:ID="Napoje">
  <rdfs:subClassOf rdf:resource="#ArtykułSpożywczy" />
  <!-- dalsze dane o klasie -->
</owl:Class>
```

Nie są to jednak stricte elementy ontologii win, ale klasy pomocnicze, którymi będziemy musieli się posłużyć jako klasami pomocniczymi. Zdaniem J. Strzelnickiego (2004), powinny one zostać zdefiniowane w osobnej ontologii, dotyczącej produktów spożywczych właśnie.

Przy definiowaniu naszej ontologii win, klasy “spożywcze” są importowane. Tak więc autor tworzy prostą definicję naszej klasy *Wino* w następujący sposób:

```
<owl:Class rdf:ID="Wino">
  <rdfs:subClassOf rdf:resource="#Napój"/>
  <rdfs:label xml:lang="en">wine</rdfs:label>
  <rdfs:label xml:lang="fr">vin</rdfs:label>
  <rdfs:label xml:lang="pl">wino</rdfs:label>
  ...
</owl:Class>
```

Poza samymi klasami, definiuje również jednostki. Najprostsza taka definicja stwierdza, że dana jednostka jest członkiem konkretnej klasy:

```
<Region rdf:ID=”RegionNadmorski” />
```

J. Strzelnicki (2004) stwierdza, że podczas budowania ontologii, ważne jest odpowiednie dobranie klas i jednostek. Klasa jego zdaniem reprezentuje typ obiektu oraz zbiór właściwości, które każdy członek tej klasy będzie posiadał. Jednostki to konkretne istniejące elementy, dające się posegregować w klasy.

Model składający się z klas i jednostek byłby zdaniem Strzelnickiego jedynie ubogą systematyką. Dlatego też posługujemy się właściwościami. Właściwości (ang. *properties*) w OWL-u można podzielić na dwie grupy:

- właściwości danych, stanowiące relacje pomiędzy klasami a typami danych XML;
- właściwości obiektów, stanowiące relacje pomiędzy elementami dwóch klas.

Definiując daną właściwość, mamy dwa sposoby na ograniczenie elementów tej relacji. Możemy podobnie ustalić dziedzinę (and. *Domain*) oraz zasięg (ang. *range*), np.:

```
<owl:ObjectProperty rdf:ID=”zrobioneZWinogron”>
  <rdfs:domain rdf:resource=”#Wino”/>
  <rdfs:range rdf:resource=”#Winogrono”/>
</owl:ObjectProperty>
```

Właściwości, podobnie jak klasy, można organizować w hierarchię:

```
<owl:ObjectProperty rdf:ID=”OpisWina” />

<owl:Class rdf:ID=”KolorWina”>
  <rdfs:subClassOf rdf:resource=”#OpisWina” />
  ...
</owl:Class>

<owl:ObjectProperty rdf:ID=”maOpisWina”>
  <rdfs:domain rdf:resource=”#Wino” />
  <rdfs:range  rdf:resource=”#OpisWina” />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID=”maKolor”>
  <rdfs:subPropertyOf rdf:resource=”#maOpisWina” />
```

```
  <rdfs:range rdf:resource=”#KolorWina” />
</owl:ObjectProperty>
```

Podobnie jak i inne języki do opisu ontologii, OWL posiada mechanizmy pozwalające na ograniczanie relacji pomiędzy obiektami w konkretnych kontekstach. Mechanizmy te pozwalają na wprowadzenie ograniczeń podobnych do zasięgu, ale w kontekście lokalnym. **AllValuesFrom** jest relacją silną, wymuszającą fakt, że wszystkie wartości danej właściwości będą określonej klasy. Natomiast **someValuesFrom** jest nieco mniej restrykcyjne, wymusza jednak to, że chociaż w jednym przypadku wartością właściwości jest element będący członkiem takiej klasy. Oto przykład obu takich relacji (za Strzelnickim, 2004):

```
<owl:Class rdf:ID=”Wino”>
  ...
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource=”#posiadaProducenta” />
      <owl:allValuesFrom rdf:resource=”#Winiarnia” />
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>
```

Innymi słowy stwierdzamy, że producentem wina musi być winiarnia, ale w przypadku innych produktów nie jest to konieczne. W dość podobny sposób możemy wpływać na tzw. liczebność. Służy do tego właściwość owl:cardinality, która precyzuje dokładną ilość elementów w relacji. Przykładowo określmy, że Rocznik będzie klasą, która może “mieć” tylko jeden Rok.

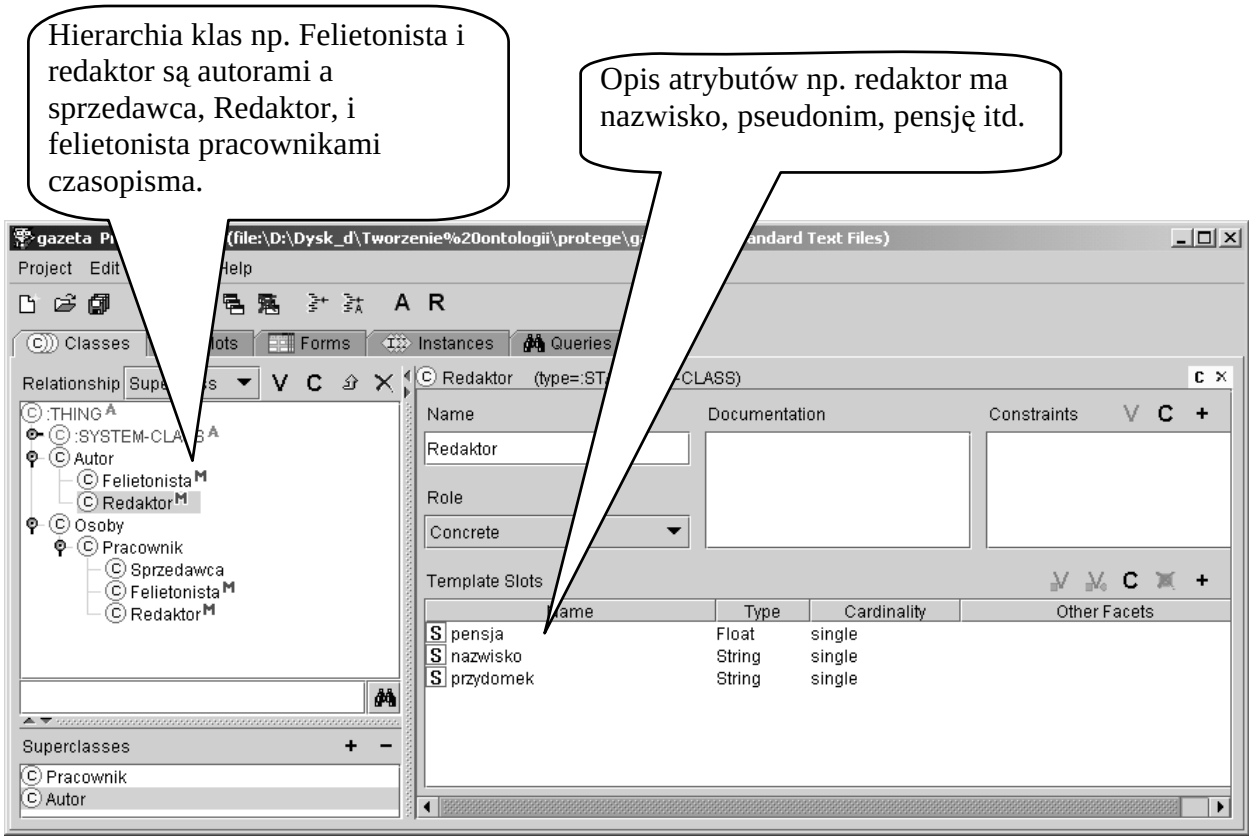
```
<owl:Class rdf:ID=”Rocznik”>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource=”#wyprodukowanyWRoku”/>
      <owl:cardinality>1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Przedstawiono dość uproszczone przykłady zastosowania języka OWL. Szczegółowe jego omówienie znajdzie czytelnik m.in. w pracach: OWL, 2004a , Bechhofer, i in., 2004, OWL, 2004b.

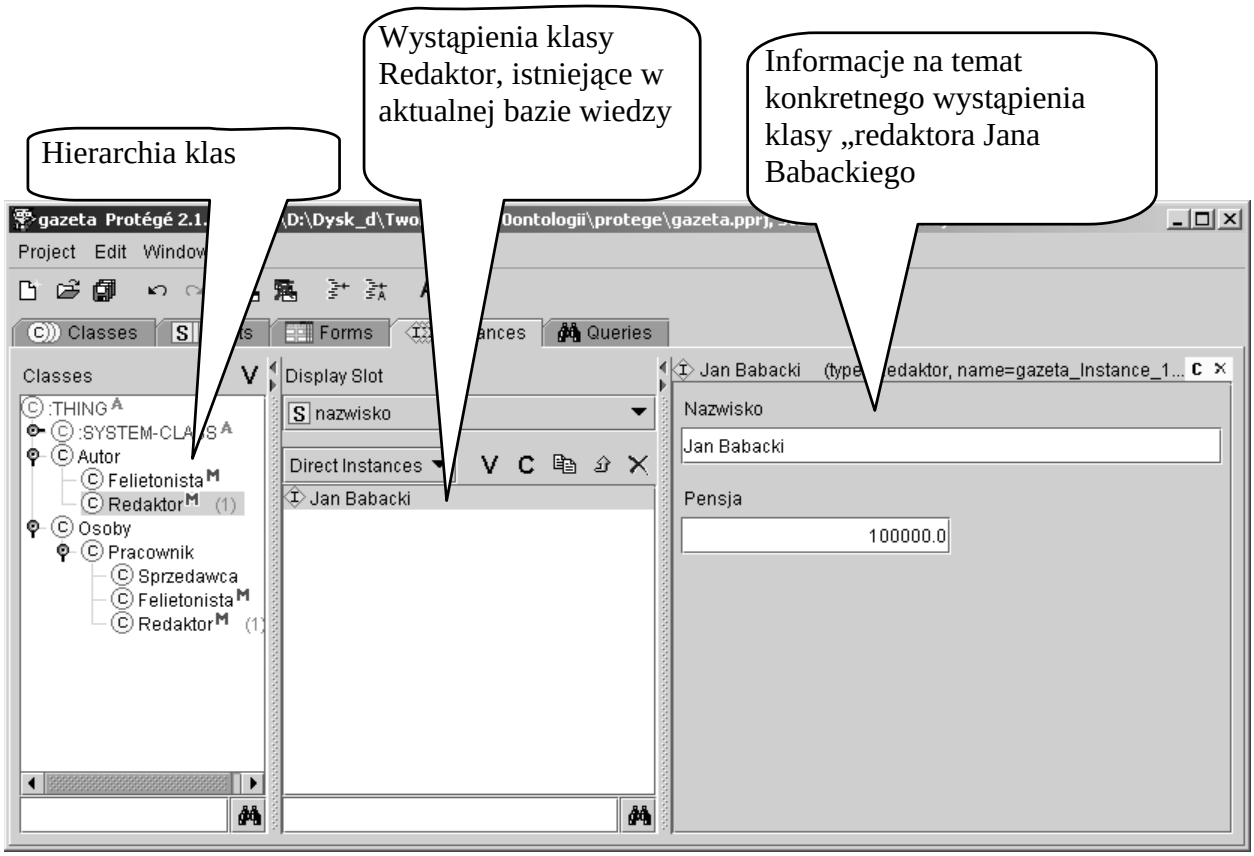
2.4. EDYTORY: PROTEGE I OIL

Protege-2000 został stworzony na uniwersytecie w Stanford w Stanach Zjednoczonych. Służy do tworzenia samych ontologii, jak również – na ich podstawie – baz wiedzy. Jest darmowym projektem typu Open Source. W OIL wszystkie klasy są klasami

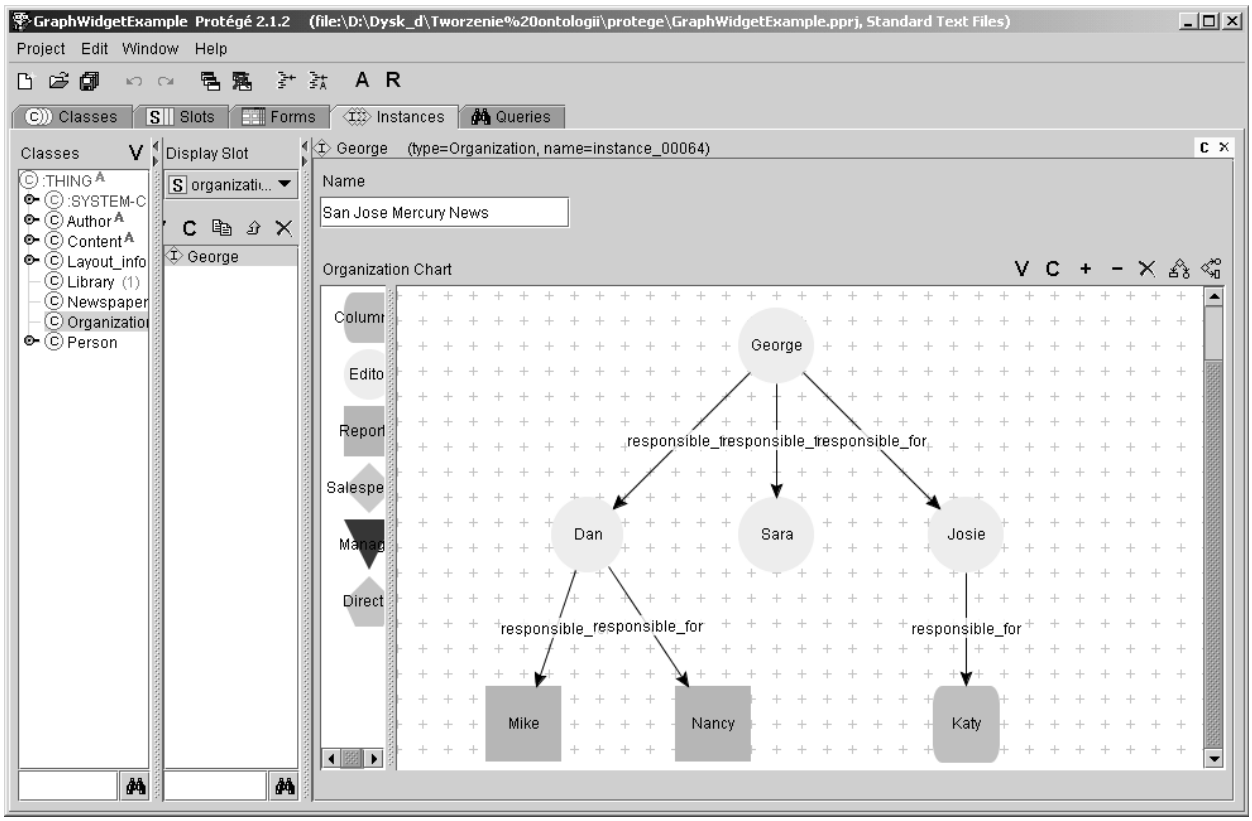
potomnymi klasy Thing, którą można jednak ukryć. Tworzone klasy prezentowane są od razu w formie hierarchii, przy czym można je przemieszczać za pomocą wskaźnika, co ułatwia organizację klas i jest jedną z podstawowych zalet programu (**Rys. 1**, **Rys. 2**). Protege pozwala również na graficzną reprezentację zależności między poszczególnymi wystąpieniami klas (**Rys. 3**). W zakładce atrybutów **Slots** definiujemy zarówno poszczególne właściwości danych klas, jak i zachodzące między nimi relacje. Tworzenie wystąpień klas jest możliwe w zakładce **Instances**. G. Wlekły zaleca rozszerzenie możliwości Protege-2000 za pomocą zainstalowania dodatkowych programów – „wtyczek" (ang. *plugin*), tworzonych przez użytkowników z całego świata. Autor zastosował dwie takie wtyczki, służące wizualizacji – TGVizTab oraz Ontviz Tab. Pierwsza okazała się skutecznym narzędziem do pokazania hierarchii klas. Druga pozwoliła na pokazanie właściwości klas oraz relacji między nimi zachodzących. Zdaniem G. Wlekłego pewną wadą Protege-2000 jest konieczność stosowania wtyczki w przypadku eksportu danych do OWL (Wlekły, 2004).



Rys. 1 Opis klas i ich atrybutów w Protege.



Rys. 2 Wystąpienia klas w Protege.



Rys. 3 Reprezentacja graficzna relacji między wystąpieniami klas w Protege.

Program OIL powstał na uniwersytecie w Manchesterze we współpracy z innymi uniwersytetami europejskimi, w ramach projektu OIL sponsorowanego przez Unię Europejską. OIL ma za zadanie stworzenie mechanizmów umożliwiających prezentację ontologii na stronach internetowych.

Jak wspomina J. Strzednicki (2004), autorzy systemu OIL starali się połączyć pracę trzech różnych niezależnych grup, zajmujących się:

– systemami ramowymi (ang. *Frame-based systems*), stosowanymi przy projektowaniu systemów sztucznej inteligencji. Ogólna koncepcja opiera się na tym, że najniższym elementem modelowania jest klasa (zwana ramką), posiadająca właściwości (zwane *slot*);

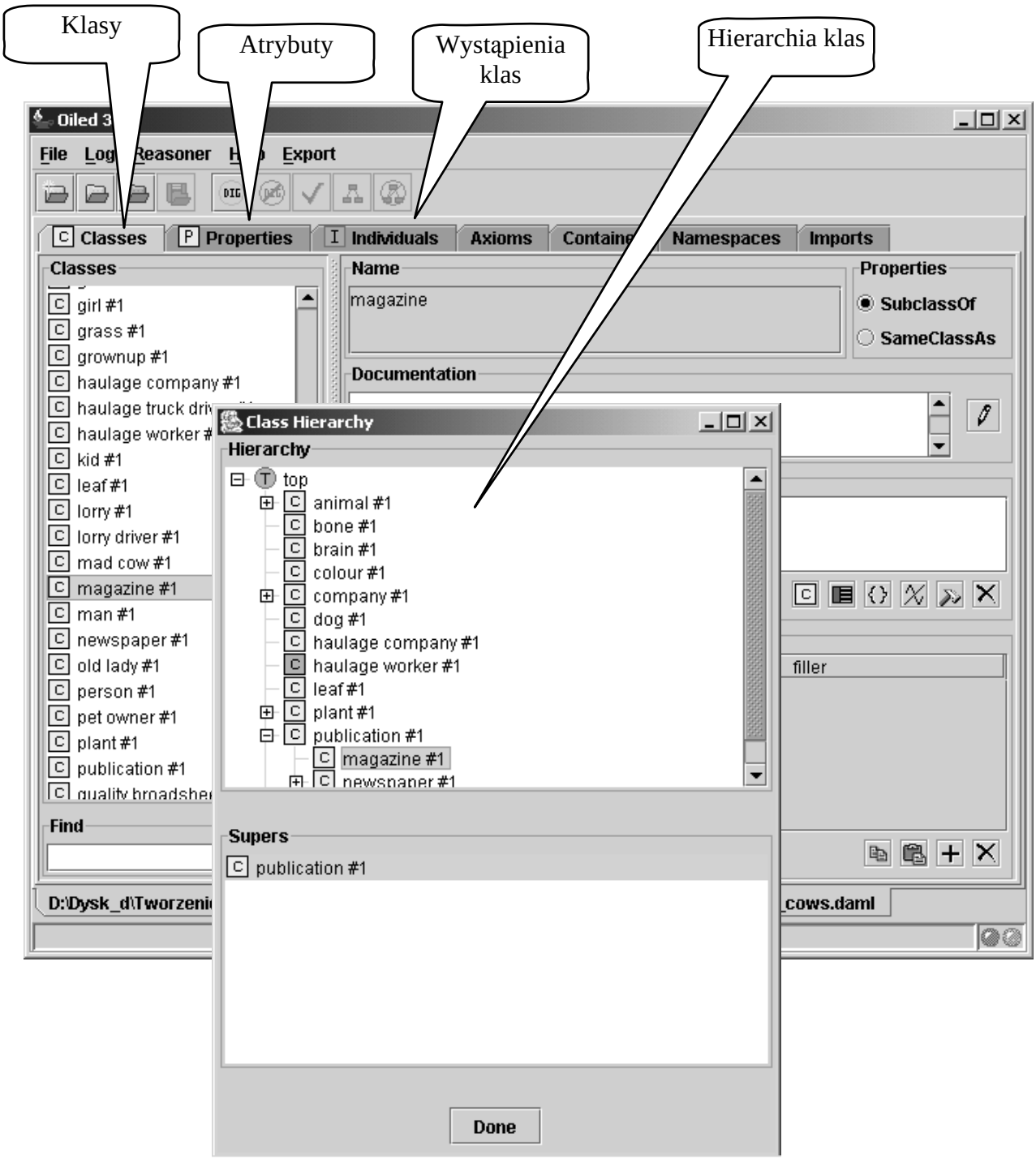
– logiką opisową (ang. *Description Logics*). Logika ta została opracowana w trakcie badań nad reprezentacjami wiedzy; opisuje wiedzę za pomocą tzw. koncepcji (które można porównać z klasami bądź też ramkami) i ról (porównywalnych ze *slotami*). Ważną rolę w logice opisowej odgrywa fakt, że posiada ona dobrze udokumentowane i opracowane podstawy teoretyczne i że każde wyrażenie może zostać przedstawione w ścisły matematyczny sposób;

– XML i RDF oraz ich rolą, jaką odgrywają w Semantycznym Webie.

System OIL jest darmowy i rozpowszechniany jest na zasadzie General Public License (GPL). Edytor OilEd powstał w ramach projektu OIL dlatego podstawowy format zapisu, który

wykorzystuje jest zgodny z językiem OIL. OilEd umożliwia także eksport do formatu RDF, DAML czy nawet HTML.

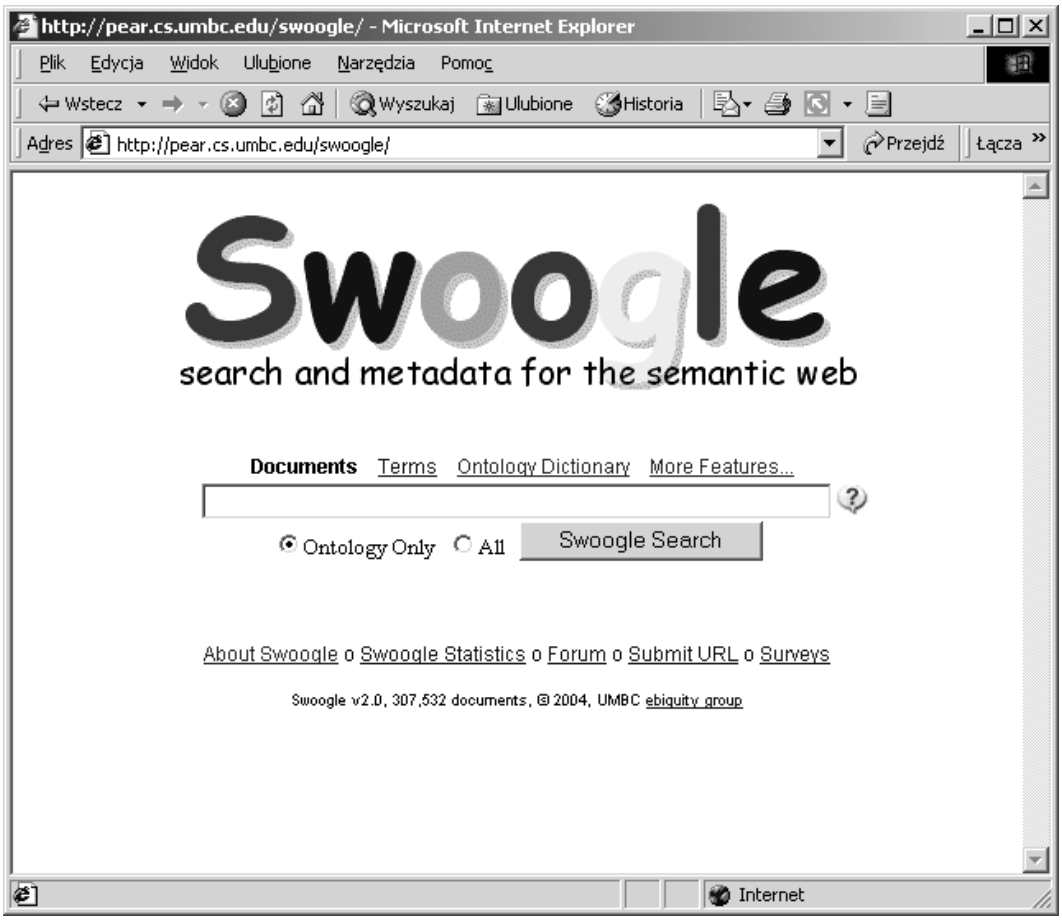
W skład edytora wchodzi narzędzie pozwalające sprawdzić zgodność logiczną ontologii. Jest to bardzo przydatne narzędzie, jednak jego przydatność wzrasta wraz ze wzrostem objętości ontologii, a OilEd nie wspiera tworzenia dużych rozwiązań. Edytor jest narzędziem zamkniętym i ewentualne dodatki muszą operować na już zapisanej ontologii, a zgodność z językiem OIL pozwala na wykorzystanie wielu istniejących narzędzi. OilEd, podobnie jak właściwie wszystkie narzędzia jest napisany w Javie. Choć interfejs jest dość czytelny, to do jego podstawowych wad należy zaliczyć brak drzewa pojęć dostępnego w zakładce programu. Drzewo można oczywiście zobaczyć w osobnym oknie, co jednak utrudnia tworzenie oraz przeglądanie ontologii. Podobnie jest z relacjami, które również są prezentowane przez otwarcie osobnego okna (**Rys. 4**).



Rys. 4 Edytor OIL.

3. SWOOGLE – WYSZUKIWARKA DOKUMENTÓW SEMANTYCZNEGO WEBA

W marcu 2004 r. pojawiła się pierwsza wersja systemu wyszukiwawczego SWOOGLE (www.swoogle.org) stworzona przez Uniwersytet w Baltimore (Maryland, Stany Zjednoczone)[1]. Obecnie istnieje już wersja druga systemu.Wykorzystuje on ontologie OWL Swoogle jest wyszukiwarką stosującą system wyszukiwawczy dla dokumentów Semantycznego Weba (DSW)[2], tzn. dokumentów zapisanych w RDF, OWL lub N3[3].



Rys. 5 Strona powitalna Swoogle

Swoogle dokonuje ekstrakcji metadanych z dokumentów i stara się ustalić relacje między nimi. Znalezione dokumenty są ponadto indeksowane przez system informacyjny,

[1] Projekt był finansowany przez DARPA F30602-00-0591 oraz NSF (National Science Foundation) NSF-ITR-IIS-0326460 and NSF-ITR-IDM-0219649.
[2] Ang. Semantic Web Document (SWD).
[3] N3 jest kolejnym językiem semenatycznego Weba po RDF, który charakteryzuje się większą ekspresywnością [NOTATION-3 (2004)].

który może wykorzystywać gramatykę N-Gram (Markovia)[4] lub odwołania URI (Uniform Resource Indetifier) jako słowa kluczowe, w celu znalezienia relewantnych dokumentów i obliczenia podobieństwa między dokumentami. Jedną z interesujących własności, jaką ma ponoć ustalać Swoogle jest stopień ważności ontologii („ranking") i dokumentów Semantycznego Weba.

Baza danych Swoogla aktualnie nie przechowuje wszystkich dokumentów Semantycznego Weba, które zostały znalezione, lecz stara się przechowywać rozszerzone metadane o dokumentach i terminach oraz obiektach, które zostały zdefiniowane i są używane. Co jest analogiczne do powszechnego sposobu przechowywania informacji o lokalizacji zbiorów. Na przykład, Google (za Swoogle, 2004) nie przechowuje całej zawartości dokumentu w swoich indeksach, ale raczej słowa, które są używane w dokumencie wraz z częstotliwością ich występowania. Jednakże dla dokumentów Semantycznego Weba istnieje znacznie więcej metadanych, np.: terminy które są zdefiniowane i wykorzystane oraz relacje z innymi dokumentami.

Baza danych Swoogla posiada informacje na temat 307 532 semantycznych dokumentów Weba, w czym zawiera się 43 162 267 uporządkowanych trójek i 95 054 klas, 52 619 własności oraz 6 825 767 indywiduów (wystąpień klas). Tylko 4162 z tych dokumentów jest ontologiami, które definiują klasy oraz własności, w przeciwieństwie do stwierdzeń o faktach dotyczących obiektów. Obecnie najbardziej popularnym typem dokumentów są zbiory FAOAF[5] oraz RSS[6].

Swoogle w założeniu ma wspomagać serwisy konieczne do obsługi agentów programowych i programów przez interfejs WWW; przeznaczony jest w większym stopniu dla badaczy semantycznego Weba niż dla przypadkowych użytkowników przeszukujących Internet.

Korzystając ze Swoogla można znaleźć wszystkie dokumenty semantycznego Weba, które wykorzystują zbiór własności lub klas, lub definiują klasy, których lokalne nazwy zawierają pewne ciągi znaków lub te, które korzystają z zewnętrznych ontologii.

Swoogle wykorzystuje maszynę IBMP615, bazę mySQL do przechowywania dokumentów, serwer Apache (jako serwer WWW) oraz serwer php jako serwer przetwarzania danych.

---

[4] Więcej informacji na temat gramatyki N-Gram znajdzie czytelnik na stronach W3C (N-GRAM, 2004a).
[5] FOAF 'friend of a friend', jest częścią Semantycznego Weba – projekt RDFWeb (FAOAF, 2004).
[6] RDF Site Summary lub popularnie zwane Really Simple Syndication to alternatywny w stosunku do standardowych stron WWW sposób dostępu do zasobów Internetu, zwłaszcza wiadomości i blogów (Miller et al., 2004).

Obecnie Swoogle oferuje 3 usługi:

– wyszukiwanie Swoogle[7], uwzględniające ograniczenia semantycznych dokumentów Weba (DSW), URL oraz klasy i własności wykorzystywane do ich definiowania;

– słownik ontologii[8], który indeksuje klasy i własności zdefiniowane przez znalezione DSW;

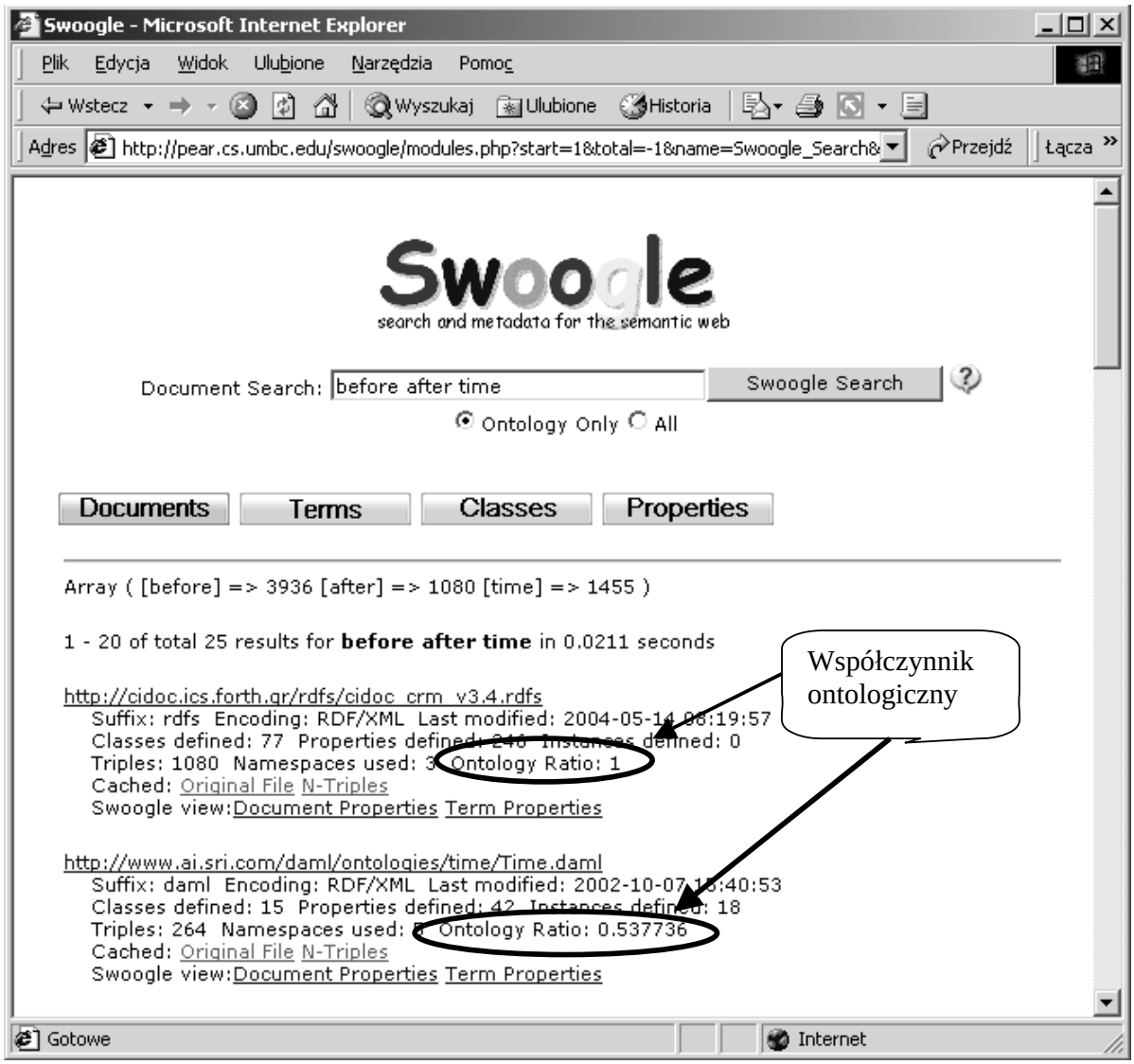
– statystyka Swoogla[9], opisująca Semantyczny Web przy wykorzystaniu metadanych zawartych w znalezionych dokumentach Semantycznego Weba.

Domyślnie znalezione dokumenty Swoogle wyświetla z tzw. URI zawierającym słowo kluczowe. Na przykład, chcąc znaleźć dokumenty o czasie „time" używamy słów kluczowych np.:

```
before after time
```

ponieważ zakładamy, że ontologie dotyczące czasu powinny zdefiniować jakieś klasy lub własności wykorzystując podane ciągi znaków.

[7] http://swoogle.umbc.edu/index.php
[8] http://swoogle.umbc.edu/modules.php?name=Ontology_Dictionary
[9] http://swoogle.umbc.edu/modules.php?name=Swoogle_Statistics



Rys. 6 Przykładowe wyszukiwanie w Swoogle.

Aby znaleźć dokumenty Semantycznego Weba zawierające termin „has parent”, należy użyć cudzysłowu. Istnieje więcej opcji wyszukiwawczych możliwych dzięki zastosowaniu prefiksów „prefiks:słowo_kluczowe”, np. jeśli chcemy znaleźć dokument Semantycznego Weba z umbc.edu w jego url piszemy:

```
url:umbc.edu
```

Domyślnie elementy składowe w zapytaniu są łączone operatorem „and” . Na przykład, chcąc znaleźć dokumenty zawierające słowo „time” jako zbiory typu N3 będące ontologiami zapiszemy

```
Time type:n3 sort:onto
```

Oczywiście użytkownicy mogą wykorzystać dowolny operator, Np., kiedy pragniemy znaleźć zbiór daml[10] lub owl, który zawiera słowo “agent”:

```
type:daml    /or    type:owl    agent
```

Tabela 2 Prefiksy wykorzystywane w wyszukiwaniu Swoogle.

Prefiks	Przykład zastosowania	Semantyka
1	**2**	**3**
„uri:”	słowa kluczowe	Wyszukiwanie słów kluczowych dla URI dokumentów. Jeśli nie zostanie określony żaden prefiks Swoogle użyje „uri:” jako przedrostka domyślnego
„type:”	“rdf”,”rdfs”,”daml”,”owl”,”rss”,”foaf”,... Można użyć dowolnego typu	Wyszukuje dokumenty przez określenie jego typu. Swoogle identyfikuje dokumenty wykorzystując ich rozszerzenia (w nazwie)
„encoding:”	„xml/rdf”, „xml”, „rdf” dla zbiorów typu xml; „n3” dla zbiorów typu n3; “n-triple”, “triple” dla zbiorów typu n-triple	Wyszukuje dokumenty wykorzystując ich sposób kodowania
„sort:”	“ontology”, “onto”, “swo”(semantic web ontology) dla zbiorów ontologii ; “instance”, “individual”, “instances”, “individuals”, “swi”(semantic web instance), “swdb”(semantic web database), “data” dla zbiorów wystąpień	Wyszukiwanie dokumentów z uwzględnieniem ich rodzaju. Swoogle wykorzystuje tzw. współczynnik ontologiczny, informujący, jak dużo ontologii zawiera dany document (por. **Rys. 6**)

Tabela 3 Operatory logiczne w Swoogle.

Operatory logiczne	
AND	„/and”, „\and”
OR	„/or”,”\or”
NOT	„-„„”/not”,”\not”
(,)	„(„„”)”

[10] Język semantycznych znaczników do opisu zasobów WWW (http://www.w3.org/TR/daml+oil-reference, http://www.w3.org/TR/daml+oil-walkthru/.

4. ZAKOŃCZENIE

Wiele już powiedziano na temat semantycznego Weba. Jednak coraz częściej słychać postulaty, aby rozpocząć jego realizację. Bezdyskusyjnym jest fakt, że ontologie są jego istotnym elementem składowym. W artykule starano się przedstawić wybrane języki oraz narzędzia pomocne w tworzeniu ontologii oraz pierwszą publicznie dostępną wyszukiwarkę dokumentów Semantycznego Weba - Swoogle. Implementacji Semantycznego Weba czy samych ontologii poświęca się coraz więcej uwagi w literaturze przedmiotu. W Polsce już w 2002 r. prowadzone były próby implementacji ontologii w korporacyjnych systemach informacyjnych dla operatora sieci telefonii cyfrowej, co opisano w pracach W. Glińskiego (2003a) oraz M. Brzozowego, Glinskiego i T. Gerszberga (2002). Skorzystano w tym celu z narzędzi OIL oraz Protege. Również ciekawe badania prowadzono ostatnio w Akademii Górniczo-Hutniczej w Krakowie na temat rejestracji przypadków wystąpień zakażeń szpitalnych w ponad 100 szpitalach z terenu całej Polski (Zygmunt i Koźlak, 2004). Zebrane informacje o pacjentach okazały się niepełne lub nadmiarowe i dlatego stworzone zostały ontologie dotyczące operacji, danych pacjenta, pobytu na oddziale, oddziałów szpitalnych., kart rejestracyjnych oraz zakażeń. Do stworzenia ontologii wykorzystano narzędzie Protege 2000, do integracji - biblioteki JENA użyte do operowania na wiedzy wyrażonej w postaci ontologicznej oraz platformę agentową JADE.

Należy z dużą nadzieją odnotować fakt, że pojawiła się już pierwsza wersja wyszukiwarki dokumentów Semantycznego Weba. Na razie jej zasięg i możliwości nie są nawet porównywalne z tradycyjnymi wyszukiwarkami internetowymi. Sceptycy uważają, że ontologie wraz z systemem odpowiednich agentów nie są w stanie wyprzeć tradycyjnych systemów wyszukiwawczych. Na obecnym etapie do pewnego stopnia jest to twierdzenie uzasadnione, ale wzrost zainteresowania Semantycznym Webem jest nie do przecenienia. Wzrasta też liczba narzędzi i platform, jak również języków do tworzenia dokumentów Semantycznego Weba. Pozostaje więc wierzyć, że wizja Tima Bernersa-Lee niedługo się sprawdzi.

LITERATURA

Bechhofer, S.; van Harmelen, F.; Hendler, J.; Horrocks, I.; McGuinness, D.L.; Patel-Schneider, P.F.; Stein, L.A. (2004). *OWL Web Ontology Language Reference W3C Recommendation 10 February 2004* [online]. W3 Consortium [dostęp: 10 05. 2005]. Dostępny w WWW: http://www.w3.org/TR/owl-ref/

Berners-Lee, T.; Hendler, J.; Lassila, O. (2001). The Semantic Web [online]. *Scientific American* [May] [dostęp: 10 05. 2005].Dostępny w WWW: http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21&pageNumber=2&catID=2

Brzozowy, M.; Glinski, W.; Gerszberg, T. (2002). Zastosowanie narzędzi lingwistycznych w wyszukiwaniu wskaźników porównawczych w kontekście operatorów sieci telefonii cyfrowej. W: *Multimedialne i sieciowe systemy informacyjne*. Materiały konferencyjne pod red. Cz. Daniłowicza. Wrocław: Oficyna Wydawnicza Politechniki Wrocławskiej, s.145-158

FAOAF. (2004). *1st Workshop on Friend of a Friend, Social Networking and the Semantic Web* [online] W3 Consortium. SWAD Europe [dostęp: 10 05. 2005] Dostępny w WWW: http://www.w3.org/2001/sw/Europe/events/foaf-galway/

Gliński, W. (2003a). Ontologies for Mobile Operators. W: *Technology for Mobile Society*. Warszawa: Wydaw. Most Press, s. 394-404

Gliński, W. (2003b). Searching the Internet for Mobile Opertors Benchmark Categories. W: *Technology for Mobile Society*. Warszawa: Wydaw. Most Press, s. 405-421.

Gliński, W. (2004). Kwestie metodyczne projektowania ontologii w systemach informacyjnych. W: *Strategie informatyzacji i zarządzanie wiedzą*. Warszawa: Wydawnictwa Naukowo-Techniczne, s. 201-212.

Miller, E. et al. (2004). *W3C RSS 1.0 News Feed Creation How-To* [online]. W3 Consortium [dostęp: 10 05. 2005]. Dostępny wWWW: http://www.w3.org/2001/10/glance/doc/howto

Nahotko, M. (2004). *Metadane. Sposób na uporządkowanie Internetu*. Kraków: Wydaw. Universytetu Jagiellońskiego, 201 [1] s.

N-GRAM. (2004). *Stochastic Language Models (N-Gram) Specification. W3C Working Draft 3 January 2001* [online] W3 Consortium [dostęp: 10 05. 2005]. Dostępny w WWW: http://www.w3.org/TR/ngram-spec/

NOTATION-3. (2004). *An RDF language for the Semantic Web. Notation 3* [online]. W3 Consortium [dostęp: 10 05. 2005]. Dostępny w WWW: http://www.w3.org/DesignIssues/Notation3.html

OWL. (2004a). *OWL Web Ontology Language Guide W3C Recommendation 10 February 2004* [online]. W3 Consortium [dostęp: 10 05. 2005]. Dostępny w WWW: http://www.w3.org/TR/owl-guide/

OWL. (2004b). *OWL Web Ontology Language Overview W3C Recommendation 10 February 2004* [online]. W3 Consortium [dostęp: 10 05. 2005]. Dostępny w WWW: http://www.w3.org/TR/owl-features/.

Srzednicki, J. (2004). *Przegląd języków i metod do opisu ontologii* [online]. [dostęp: 10 05. 2005]. Dostępny w WWW: http://violent.dream.vg/studia/tai/semantic_web.html

Swoogle. (2004). Swoogle. Search and metadata for the semantic web [online]. UMBC [dostęp: 10 05. 2005]. Dostępny w WWW: http://pear.cs.umbc.edu/swoogle/about.php.

Wlekły G. (2004) Narzędzia do tworzenia ontologii [online]. *Gazeta IT* nr 10(29) [dostęp: 10 05. 2005]. Dostępny w WWW: http://www.gazeta-it.pl/zw/git21/narzedzia_do_tworzenia_ontologii.html.

Zygmunt, A.; Koźlak, J. (2004). Metody efektywnego zarządzania wiedzą reprezentowaną w postaci ontologii. W: *Strategie informatyzacji i zarządzanie wiedzą*. Warszawa: Wydaw. Naukowo-Techniczne, s.373-383.

ABSTRACT

Selected languages for ontology design were presented and illustrated by the examples. Two most popular ontology editors were presented: Protege and Oil. Last but not least there was presented the Swoogle – a crawler-based indexing and retrieval system for the Semantic Web -- RDF and OWL documents encoded in XML or N3. In the conclusions we discussed growing popularity of using ontologies in Poland.