STUDIA INFORMATICA 2012
Volume 33 Number 2B (106)

Krzysztof CZAJKOWSKI, Tomasz TRELA
Politechnika Krakowska, Instytut Teleinformatyki, Wydział Fizyki, Matematyki i Informatyki

ZASTOSOWANIE PULPITÓW SEMANTYCZNYCH W URZĄDZENIACH MOBILNYCH

Streszczenie. Artykuł omawia problematykę wymiany informacji pomiędzy urządzeniami mobilnymi i systemami PIM na platformie desktopowej, wykorzystującymi semantyczny pulpit. W artykule przeanalizowano i omówiono istniejące ontologie, pozwalające opisywać dane dostępne w telefonie. Zaprezentowano przykładowy program implementujący dwustronną komunikację pomiędzy systemem Nepomuk i aplikacją na platformę Google Android.

Słowa kluczowe: sieć semantyczna, pulpit semantyczny, ontologia, środowisko mobilne, platforma mobilna, smartfon

APPLICATION OF SEMANTIC DESKTOPS IN MOBILE DEVICES

Summary. The article discusses the issues of information exchange between mobile devices and PIM applications on desktop platform, which uses Semantic Desktop technology. The article describes existing ontologies for representation data on smartphone platform. Article presents example program, which implements both way communication between Nepomuk System and Google Android application.

Keywords: semantic web, semantic desktop, ontology, mobile ontology, smartphone, mobile platform

1. Wstęp

Szybki rozwój technologii mobilnych spowodował, że telefony komórkowe, a w szczególności smartfony, udostępniają funkcje dostępne dotychczas wyłącznie w programach PIM (Personal Information Management) na platformie desktopowej. Semantic Desktop [1], czyli semantyczny pulpit, ma na celu gromadzenie wiedzy użytkownika w jednym miejscu, aby ułatwić zarządzanie wiedzą i łatwą wymianę informacji z innymi aplikacjami opartymi na

sieciach semantycznych. Niniejszy artykuł jest poświęcony zweryfikowaniu obecnych standardów sieci semantycznych pod kątem możliwości ich zastosowania przy tworzeniu aplikacji mobilnej, pozwalającej udostępniać informacje dla semantycznego pulpitu.

2. Pulpit semantyczny

W dobie niezwykle dynamicznego rozwoju Internetu oraz nieustannie zwiększającej się ilości danych, dużym wyzwaniem staje się opracowywanie rozwiązań mających usprawnić przeszukiwanie oraz zarządzanie informacją. Jednym z możliwych podejść w tym temacie jest stosunkowo nowe rozwiązanie, czyli Semantic Web (Sieć Semantyczna) [24]. Jest to obszar licznych prac badawczych oraz opracowywanych podejść i implementacji [25, 26].

Jednym z kierunków badań jest Semantic Desktop (pulpit semantyczny). Semantic Desktop jest to termin odnoszący się do oprogramowania wykorzystującego technologie Semantic Web w zastosowaniach desktopowych, tj. w oprogramowaniu komputerów osobistych. Idea semantycznego pulpitu nie jest nowa i sięga nawet 1945 roku, jak opisuje Leo Sauermann [2], ale termin „Semantic Desktop" został sformułowany przez Stefana Deckera i rozwinięty przez Leo Sauermanna dopiero w 2003 roku, aby w reszcie stworzyć poniższą definicję semantycznego pulpitu: „Pulpit semantyczny to urządzenie, na którym użytkownik przechowuje wszystkie swoje informacje cyfrowe, takie jak dokumenty, multimedia i wiadomości. Są one zasobami semantycznej sieci, z których każdy reprezentowany jest przez unikalny identyfikator URI. Wszystkie dane są dostępne i przeszukiwalne jako grafy RDF. Zasoby z sieci mogą być przechowywane, autoryzowana zawartość może być współdzielona z innymi osobami. Ontologie pozwalają użytkownikowi wyrażać osobiste relacje i formować semantyczny klej łączący informacje z systemami. Aplikacje przechowują, czytają i komunikują się poprzez protokoły sieci semantycznych. Pulpit semantyczny jest suplementem pamięci użytkownika." [2]

Idea semantycznego pulpitu została zaimplementowana między innymi w projekcie NEPOMUK [3]. Celem systemu Nepomuk jest zarządzanie osobistymi danymi oraz współdzielenie ich z innymi użytkownikami. System posiada mechanizmy przeszukiwania zasobów komputera osobistego, takich jak: klient poczty, kontakty z książki adresowej, zakładki z przeglądarek internetowych, a także analizuje treść przeszukanych plików, automatycznie proponując słowa kluczowe, które następnie mają ułatwić kategoryzowanie danych.

Kolejnym z obszarów zastosowań technologii semantycznych jest praca z danymi na komputerze osobistym. Badania przeprowadzone przez Bergmanna [4] udowodniły, że podstawowym działaniem użytkowników jest praca z plikami. Większość użytkowników (68%) preferuje metodę manualnego przeglądania plików (przechodząc do danego katalogu) aniżeli

wyszukiwania plików. Udoskonalone silniki szukające (np. Google Search Desktop, Spotlight) zwiększyły skuteczność szukania, ale jednakże nie zastąpiły wybierania pliku przez przeglądanie. Użytkownicy preferują wybieranie pliku, ponieważ pamiętają jego lokalizację w strukturze, którą sami tworzą. Jednakże problem pojawia się, gdy dany plik może pasować do wielu folderów, wtedy efektywność tego podejścia jest bardzo mała. Ten problem rozwiązują inteligentne wyszukiwarki, wykorzystujące metadane do indeksowania plików. Przykładem jest Strigi [5], który jest systemem wyszukiwania w KDE 4 (środowisku graficznym dla systemów Unix i GNU/Linux – K Desktop Environment). Strigi pozwala wyszukiwać dane w bardzo złożony sposób, zwracając trafne wyniki. W połączeniu z innymi narzędziami, jak Soprano i Nepomuk, KDE 4 wdraża prawdziwie semantyczny pulpit.

3. Zastosowania Semantic Desktop w środowisku mobilnym

3.1. Ontologie mobilne

Ontologie mobilne są istotnym zagadnieniem badanym w tym opracowaniu. Ontologie te są kluczowym elementem potrzebnym do modelowania i wymiany danych w aplikacji mobilnej, tworzonej w części praktycznej. Badania przeprowadzone w ramach niniejszej pracy nad ontologiami wykazały istnienie kilku ontologii próbujących objąć temat technologii mobilnej.

Autorzy „Mobile Ontology" [7] (MO) stworzyli ontologię do reprezentacji urządzeń mobilnych. MO skupia się wokół samych urządzeń, systematyzuje i formalizuje wiedzę z zakresu: parametrów sprzętowych telefonu, takich jak parametry sprzętowe, zainstalowanego firmware'u, rodzaju wspieranych stemów (GSM, CDMA, 3G, 4G), parametrów wyglądu (kolor, kształt), informacji o producencie oraz listy „stanów", jakie telefon może osiągać (np. nie może się uruchomić, błąd klawiatury, brak dźwięku).

MO nie podejmuje jednak zagadnienia, jakie informacje i działania urządzenie mobilne może wykonać. Jest zatem niewystarczająca, aby umożliwić agentom semantycznym inteligentną wymianę informacji. Jest natomiast wystarczająca, aby dokładnie zidentyfikować sam telefon.

3.2. Urządzenia przenośne i wszechobecne

W temacie urządzeń mobilnych nie sposób pominąć zagadnienia aplikacji świadomych kontekstu CWA (ang. *Context Aware Application*), działających zazwyczaj na urządzeniach przenośnych lub wbudowanych, które dostosowują swoje funkcje do aktualnego kontekstu, w jakim się znajdują. Ich szeroko rozpowszechniony rozwój sprowokował badania nad wyko-

rzystaniem w tym zakresie sieci semantycznych. Ideę CWA oraz zyski płynące z jej potencjalnego zastosowania prezentuje Ora Lasilla [8]. Obecnie używane rozwiązania opierają się na sztywno zapisanych regułach (np. dostępne „akcje" po podłączeniu telefonu do komputera PC), które mimo wszystko przyczyniły się do tzw. Interoperability Nightmare. Natomiast Lasilla proponuje wykorzystanie technologii semantycznych do stworzenia kontekstu i wykorzystanie go w następujących obszarach:

- pozyskiwania informacji,
- interfejsu użytkownika,
- wykrywania usług,
- bezpieczeństwa i prywatności,
- automatyzacji.

Znajomość kontekstu pozwoliłaby aplikacjom CWA dostosować się do aktualnych potrzeb lub możliwości użytkownika, w szczególności przez dostarczanie informacji i wskazywanie takich aspektów aplikacji, jak np. jej dostępność lub niedostępność w danym momencie, działania możliwe do wykonania bez interakcji użytkownika.

Podczas badań nad ontologiami z dziedziny urządzeń mobilnych znaleziono liczne ontologie m.in. CC/PP [11], COBRA-ONT [10], CoDAMoS [12], Delivery Context [13], SOUPA [14], które zostały zakwalifikowane przez Poveda-Villalón, Suárez-Figueroa, García-Castro, Gómez-Pérez [15] w następujący sposób (tabela 1):

Tabela 1

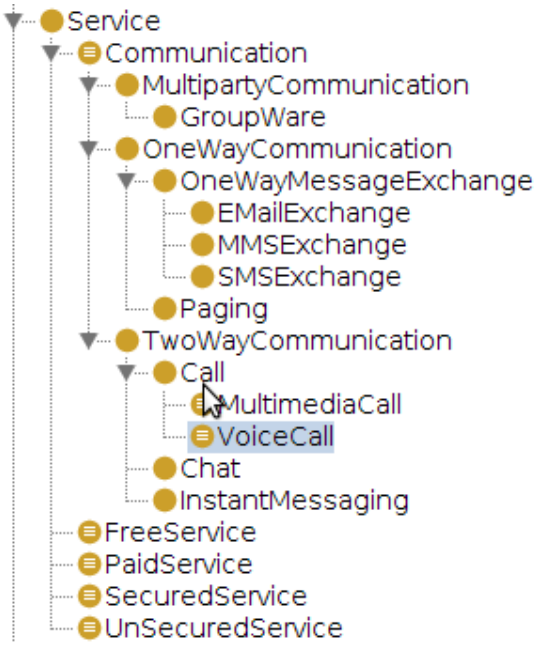
Ontologie dla urządzeń mobilnych

Ontologia / Dziedzina	CC/PP	COBRA-ONT	CoDA-MoS	Delivery context	SOUPA
Urządzenie	x	x	x	x	
Środowisko		x	x	x	
Interfejs					
Położenie		x	x	x	x
Sieć				x	
Operator					
Rola		x	x		
Usługa			x		
Źródło					
Czas		x	x		x
Użytkownik	x	x	x		x

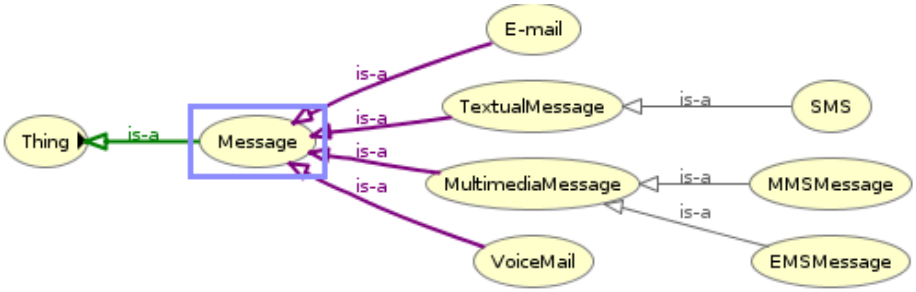
Jak widać, wiele ontologii opisuje wybrane zagadnienia z dziedziny telekomunikacji, ale żadna z powyższych nie reprezentuje usług dostępnych w telefonie. Za ich pomocą nie można opisać danych, takich jak ostatnio wykonane połączenie czy lista odebranych SMS.

3.3. Ontologia telekomunikacyjna

W tym rozdziale krótko zaprezentowano ontologię stworzoną w ramach projektu SIMS (Semantic Interfaces for Mobile Services) [16]. Z punktu widzenia niniejszego opracowania, ontologia ta jest interesująca, ponieważ opisuje dziedzinę wybraną do analizy w części praktycznej. Celem utworzenia SIMS Ontology [17] było dostarczenie dziedzinowej terminologii na potrzeby tworzenia aplikacji opartych na usługach mobilnych opisanych za pomocą ontologii. Okazuje się, że ontologie SIMS kompleksowo segregują i definiują pojęcia związane z usługami komunikacyjnymi na telefonie komórkowym. W części praktycznej została podjęta próba stworzenia interfejsu wymiany danych pomiędzy urządzeniem mobilnym oraz innym system semantycznym. Z tego punktu widzenia istotna była reprezentacja takich konceptów, jak rozmowa telefoniczna, SMS, MMS. Rysunek 1 prezentuje hierarchię usług opisanych w ontologii SIMS, a rys. 2 prezentuje klasy wiadomości w ontologii SIMS.

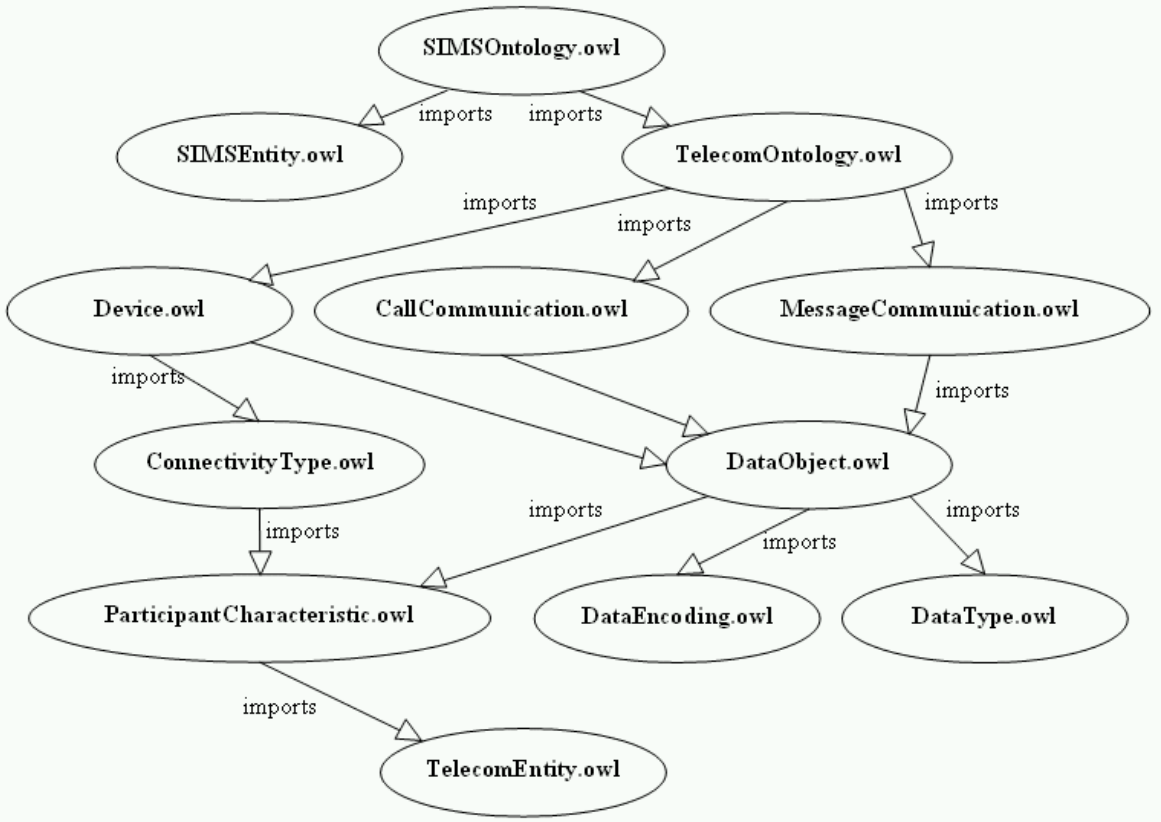


Rys. 1. Hierarchiczny model usług zdefiniowanych w ontologii SIMS
Fig. 1. Hierarchical model of services defined in SIMS Ontology



Rys. 2. Klasy wiadomości w ontologii SIMS
Fig. 2. Classes of message in the SIMS Ontology

Moduły ontologii SIMS opisują wiedzę domenową z zakresu telekomunikacji. Wyszczególnione w ontologii SIMS moduły prezentuje rys. 3:



Rys. 3. Ontologie SIMS
Fig. 3. SIMS ontologies

Poszczególne ontologie (moduły) tworzące SIMS [23]:

- Telecom top ontology (TelecomEntity.owl) – definiuje ogólne pojęcia; wykorzystywana przez inne ontologie.
- Data type (DataType.owl) – definiuje pojęcia związane z typami danych; opiera się na taksonomii MIME, odnoszącej się do typów danych zarejestrowanych przez IANA (Internet Assigned Numbers Authority).
- Communication participant (ParticipantCharacteristic.owl) – zawiera pojęcia umożliwiające szczegółowy opis uczestników komunikacji, np. identyfikacja typu adresu, stanu uczestnika.
- Data encoding (DataEncoding.owl) – definiuje pojęcia związane z kodowaniem danych, np. kodowanie tekstu, dźwięku i wideo; zawiera tylko podstawowe pojęcia, nie odnosi się do szczegółowych właściwości kodowania.
- Device (Device.owl) – definiuje pojęcia umożliwiające opis terminali użytkownika i ich możliwości; szczególnie dotyczy to telefonów komórkowych.
- Call communication (CallCommunication.owl) – zawiera pojęcia związane z połączeniami peer to peer, np. rozmowy głosowe.

- Message communication (MessageCommunication.owl) – zawiera pojęcia związane z komunikacją opartą na wysyłaniu wiadomości, np. SMS, email.
- Connectivity types (ConnectivityType.owl) – definiuje typy połączeń wykorzystywanych w komunikacji i dostępne usługi doręczyciela (podstawowe usługi telekomunikacyjne oferujące możliwość transmisji).
- Data object (DataObject.owl) – definiuje pojęcia związane z porcjami danych – obiektami danych.
- Telecom ontology (TelecomOntology.owl) – integruje wszystkie moduły w jedną ontologię przez import tych modułów oraz wprowadzenie relacji pomiędzy pojęciami w nich występującymi; sam moduł nie definiuje nowych pojęć.
- SIMS core concepts (SIMSEntity.owl) – definiuje pojęcia SIMS i relacje pomiędzy nimi.
- SIMS telecom (SIMSOntology.owl) – ontologia integrująca, importująca SIMSEntity.owl i TelecomOntology.owl oraz wprowadzająca związki pomiędzy zawartymi w nich pojęciami.

4. Aplikacja

4.1. Założenia

W celu zaprezentowania możliwości pulpitu semantycznego wykonano aplikację na platformę Google Android. Jest to platforma wykorzystywana w telefonach komórkowych, oparta na jądrze Linuksa. Programy przygotowywane na platformę Android tworzone są w języku Java.

Głównym celem jest pokazanie, jak można wymieniać się informacją za pomocą technologii semantycznych. Jedną z zasadniczych motywacji projektu NEPOMUK było umożliwienie gromadzenia informacji w jednym miejscu. System Nepomuk nie posiadał jednak bezpośredniej możliwości komunikowania się z urządzeniem mobilnym.

Przykładowa aplikacja działająca w telefonie komórkowym pobiera dostępne informacje, takie jak: lista kontaktów, numery telefonów, wysłane wiadomości i zapisane notatki. Na podstawie tych informacji program tworzy model RDF, reprezentujący te dane. Model ten jest serializowany do formatu N3, który możemy importować w systemie NEPOMUK.

Jednym z celów programu, opracowanym w części praktycznej, jest umożliwienie wymiany informacji pomiędzy urządzeniem mobilnym a systemem Nepomuk. System Nepomuk, opisany w poprzednim rozdziale, ułatwia zarządzanie wiedzą (ang. *knowledge managment*). Rozszerzenie jego zakresu o informacje dostępne w telefonie umożliwiłoby użytkownikowi dostęp np. do informacji o wykonanych rozmowach telefonicznych z danym użytkownikiem.

Zadaniem było stworzenie programu generującego informację w postaci RDF, a także przeanalizowanie, jakie typy informacji system Nepomuk potrafi gromadzić i prezentować, a w razie potrzeby je rozszerzyć. Celem była również próba rozwiązania problemów pojawiających się z przetwarzaniem danych w postaci RDF, zaimplementowanie mechanizmu wnioskowania opartego na znanych ontologiach i wyrównywaniach danych (ang. *data alignement*).

Drugą funkcją programu jest pobieranie informacji z systemu Nepomuk o zasobach dostępnych w telefonie, np. kontaktach, oraz prezentacja tych dodatkowych informacji użytkownikowi. Umożliwia to użytkownikowi łatwy dostęp do informacji znajdujących się w systemie Nepomuk bezpośrednio z telefonu komórkowego. Należy zwrócić uwagę, że system Nepomuk gromadzi informacje z wielu źródeł danych, np. książki adresowej programu MS Outlook, dlatego możliwość odczytywania tych informacji z zewnątrz jest kolejnym udogodnieniem.

Innym przykładem zastosowania funkcji drugiej aplikacji jest stworzenie notatki powiązanej z kontaktem w systemie Nepomuk, a następnie wyświetlenie jej przy odpowiednim kontakcie z książki adresowej telefonu komórkowego.

4.2. Środowisko uruchomieniowe i wykorzystywane biblioteki

Program został napisany w języku Java, dlatego możliwe jest jego uruchomienie na platformach Linux, Windows oraz w systemie OS X. Niezbędnym oprogramowaniem jest Java 6, Android SDK w wersji API 8.0, serwer Tomcat 6.0. Jako środowisko programistyczne używane podczas pisania programu wykorzystano IDE Eclipse 3.6.

Aplikacja komunikuje się z systemem Nepomuk, który można pobrać ze strony projektu [18]. W niniejszym rozwiązaniu użyta została wersja DFKI Nepomuk, która w praktyce okazała się jednak niekompletna. Dlatego konieczne stało się pobranie plików źródłowych oraz zbudowanie własnej wersji aplikacji.

Jednym z problemów jest wybór odpowiednich bibliotek umożliwiających pracę z RDF. Ponieważ aplikacja kliencka przeznaczona jest na platformę mobilną, dużym ograniczeniem są dostępne zasoby sprzętowe. Zarówno wykorzystanie pamięci, jak również intensywne obliczenia powinny być minimalne. W chwili tworzenia aplikacji biblioteka Jena nie została przeniesiona na system Android, dlatego przesyła dane do serwera pośredniczącego jako obiekty Java (POJO), a następnie serwer tworzy model RDF, korzystając z biblioteki Jena, i przesyła go do sytemu Nepomuk.

W projekcie wykorzystano bibliotekę open source Jena [19] w wersji 2.6.3, jako bazę danych RDF. Jena pozwala przechowywać trójki RDF i manipulować nimi, wspiera standardy

RDF, RDFS, OWL, SPARQL oraz GRDDL. Jest to obecnie najbogatsza biblioteka RDF napisana w języku Java.

Manipulacja modelem RDF, korzystając z biblioteki Jena, pozwala tworzyć model dzięki intuicyjnemu API.

4.3. Informacje dostępne w telefonie z systemem Android

Pierwszym krokiem jest stworzenie modelu danych, które miałyby zostać przesłane do systemu Nepomuk. Następnie wybierane są najważniejsze informacje dostępne w telefonie:

- lista kontaktów,
- historia wykonanych połączeń,
- lista zakładek przeglądarki,
- treść wiadomości wysłanych do danego użytkownika,
- aktualna informacja o położeniu, jeśli telefon jest wyposażony w czujnik GPS.

System Android udostępnia dane użytkownika za pomocą usług (ang. *content providers*), które umożliwiają bezpośredni dostęp do wielu informacji bez interakcji użytkownika.

Chcąc utworzyć model RDF, należy dokładnie przeanalizować, jakie informacje mają należeć do modelu, a następnie zbadać, za pomocą jakich ontologii można te informacje reprezentować.

Wybrane informacje zostały zgrupowane następująco:

- Informacje o kontaktach:
  - nazwa kontaktu,
  - numer telefonu,
  - lista adresów email posiadanych przez kontakt,
  - notatki skojarzone z kontaktem,
  - ilość połączeń z kontaktem,
  - data ostatniego połączenia,
  - znacznik, czy kontakt jest oznaczony gwiazdką.
- Informacje o zakładkach:
  - adres URL strony WWW,
  - data utworzenia,
  - tytuł zakładki.
- Informacje o wykonanych połączeniach:
  - data wykonania połączenia,
  - czas trwania połączenia,
  - rodzaj wykonanego połączenia (wykonane, odebrane, nieodebrane),
  - rozmówcy połączenia (numer telefonu lub nazwa kontaktu, jeśli istnieje).

- Informacje o wymienionych wiadomościach:
  - data wysłania/odebrania wiadomości,
  - zawartość wiadomości,
  - numer oraz ,jeśli istnieje, ID kontaktu adresata/nadawcy wiadomości,
  - ID wątku, do którego należy wiadomość,
  - czy wiadomość została przeczytana,
  - status wiadomości.

4.4. Wybór URI dla eksportowanych zasobów

Chcąc poprawnie utworzyć model RDF, trzeba się zastanowić, jakie URI zastosować dla zasobów. Muszą być one unikalne. W przypadku zasobów pochodzących z telefonu komórkowego, rozsądnym pomysłem jest korzystanie z numeru telefonu, z jakiego one pochodzą. To podejście nie rozwiązuje jednak problemu zmiany numeru telefonu. Innym rozwiązaniem jest wykorzystanie numeru IMEI [20]. Jest to unikalny identyfikator urządzenia CDMA, np. wykorzystywany przez operatorów niektórych sieci do blokowania skradzionych telefonów.

Numer IMEI posiada format AA-BBBBBB-CCCCCC-EE.

Ponieważ emulator telefonu Android nie pozwala na poprawne ustawienie numeru IMEI, w przykładowej aplikacji zastosowano prosty schemat URI, oparty na numerze telefonu:

semdesk://48506977506/contacts/1

System Nepomuk stosuje podobny schemat URI, ale wykorzystuje adres email użytkownika:

semdesk://jan.kowalski@example.com/things/

4.5. Reprezentacja wybranych informacji w RDF

Informacje o osobach są przechowywane w systemie Nepomuk w postaci RDF. Nepomuk korzysta z wielu ontologii. Szczególnie istotne ontologie, zdefiniowane specjalnie na potrzebny Semantic Desktop, to:

- NAO – Nepomuk Annotation Ontology – zawiera podstawowe właściwości (ang. *properties*), pozwalające m.in.: tworzyć adnotacje, relacje, tworzyć tagi, zapisać daty utworzenia czy modyfikacji. Przykładowe właściwości wykorzystane z ontologii NAO to:

  nao:isRelated, nao:hasTopic, nao:lastModified, nao:created
- NIE – Nepomuk Information Element jest zbiorem ontologii definiujących słowniki do opisu zasobów, takich jak: pliki, kontakty, wiadomości (NIE core, Nepomuk File Ontology, Nepomuk Contact Ontology, Nepomuk Message Ontology, Nepomuk Calendar Onology).

- PIMO – Personal Information Model Ontology jest główną ontologią wykorzystywaną w systemie Nepomuk. Każdy obiekt istniejący w Nepomuku jest instancją pimo:ClassOrThing.

Powyższe ontologie pozwalają utworzyć model kontaktu w RDF w następującej postaci:

```
<semdesk://48506977506/contacts/1> a pimo:Person ;
   rdfs:label "Jan Kowalski" ;
   pimo:hasOtherRepresentation <http://dbpedia.org/resource/Jan_Kowalski> ;
   pimo:wikiText """To jest mój manager""" ;
   nao:numericRating "8" ;
   nao:modified "2010-08-22T05:32:36"^^xsd:dateTime  ;
   nao:created "2010-08-22T05:32:34.429-07:00"^^xsd:dateTime ;
   nao:prefLabel "Jan Kowalski" ;
   nao:hasSymbol <bookmark>;
   nco:hasPhoneNumber "0048506977506";
   nco:emailAddress "jan.kowalski@gmail.com".
```

Zakładki mogą być reprezentowane w następujący sposób:

```
<semdesk://48506977506/bookmark/1> a pimo:Topic, nfo:Bookmark ;
   rdfs:label "Onet.pl - Polski Portal Internetowy" ;
   pimo:groundingOccurrence <http://www.onet.pl/> ;
   nfo:bookmarks <http://www.onet.pl/> ;
   nao:created "2010-08-22T09:49:58.425-07:00"^^xsd:dateTime ;
   nao:prefLabel "Onet.pl - Polski Portal Internetowy" .
```

Do reprezentacji zakładek wykorzystano istniejące ontologie, które są w tym przypadku wystarczające. Opisana wcześniej ontologia SIMS spełnia wymagania, aby reprezentować wybrane informacje o wykonanych rozmowach, takie jak:

- Call – wykonane połączenie, posiadające następujące właściwości:
  - czas połączenia,
  - godzina rozpoczęcia połączenia,
  - inicjator połączenia: numer telefonu lub kontakt posiadający przypisany numer telefonu,
  - odbiorca połączenia: numer telefonu lub kontakt posiadający przypisany numer,
  - miejsce połączenia, jeśli telefon posiada GPS,
  - typ połączenia (wykonane, odebrane, nieodebrane).
- SMS – wiadomość sms:
  - adresat, numer telefonu lub kontakt z przypisanym numerem,
  - nadawca, numer telefonu lub kontakt z przypisanym numerem,
  - czas wysłania wiadomości,
  - potwierdzenie odebrania wiadomości.

Z badań wynika, że brakuje właściwości do reprezentacji:

- liczby połączeń z kontaktem,
- daty ostatniego połączenia,

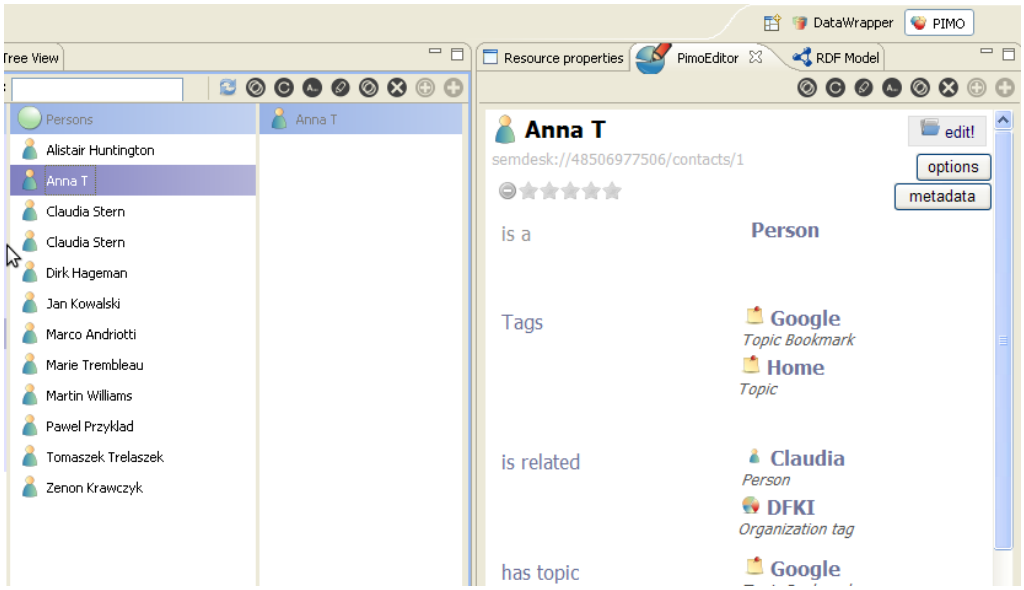
jednakże te informacje można łatwo uzyskać, korzystając z języka zapytań SPARQL.

4.6. Implementacja w języku Java

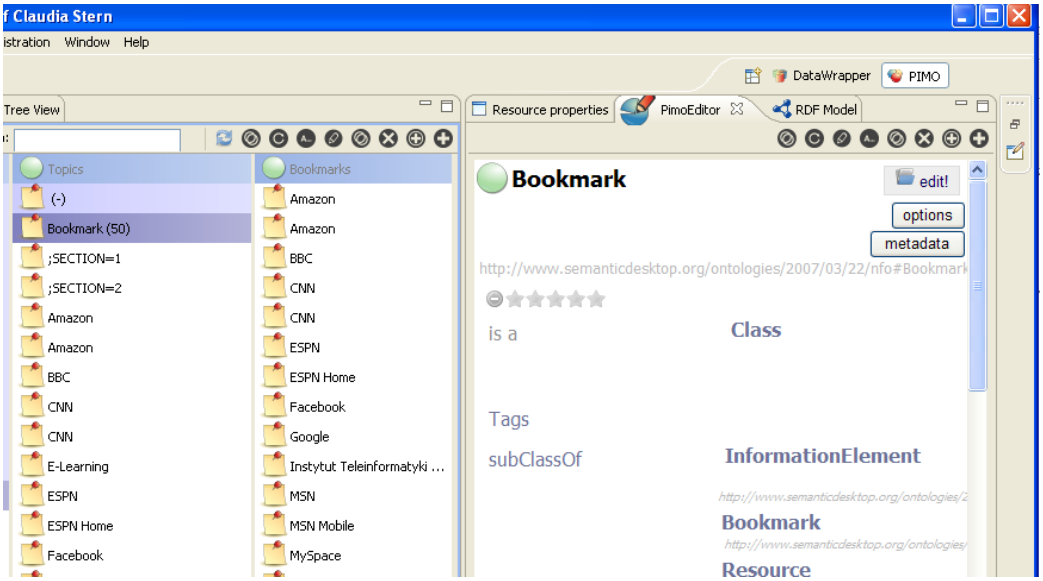
Do stworzenia modelu RDF w języku Java korzystamy z wcześniej opisanej biblioteki Jena. API biblioteki jest bardzo intuicyjne. Graf RDF jest reprezentowany przez klasę com.hp.hpl.jena.rdf.model.Model. Na modelu możemy pracować przez bezpośrednie dodawanie/usuwanie zdań RDF (ang. *statement*) lub, bardziej wygodnie, tworząc tzw. Resource (podmiot) i dodając do niego właściwości. Poniższy listing pokazuje, jak stworzyć model RDF, reprezentujący dane o zakładkach użytkownika.

```
package com.trela.server;
import ...

public class GenerateRDF {
    ...
    public static String fromData(Data data) {

        // create an empty Model
        Model model = ModelFactory.createDefaultModel();
        List<Bookmark> bookmarks = data.bookmarks;

        for (Bookmark b : bookmarks) {
            Resource bookmarkResource = model
                    .createResource("semdesk://48506977506/bookmarks/" + b.id);
            bookmarkResource.addProperty(RDFS.label, b.title);
            bookmarkResource.addProperty(RDF.type, model
                    .createResource(Vocabularies.PIMO + "#Topic"));
            bookmarkResource.addProperty(RDF.type, model
                    .createResource(Vocabularies.NFO + "#Bookmark"));
            bookmarkResource.addProperty(
                    model.createProperty(Vocabularies.NAO,
                    "prefLabel"), b.title);
            bookmarkResource.addProperty(model.createProperty(
                    Vocabularies.PIMO, "groundingOccurence"), b.url);
            bookmarkResource.addProperty(model.createProperty(Vocabularies.NFO,
                    "bookmarks"), b.url);
            if (!b.craeted.equals("0")) {
                SimpleDateFormat sdf = new SimpleDateFormat(
                        "yyyy-MM-dd HH:mm:ss.000");

                Date resultdate = new Date(Long.valueOf(b.craeted));
                String created = sdf.format(resultdate);
                bookmarkResource.addProperty(model.createProperty(
                        Vocabularies.NAO, "created"), created);
            }
        }
        StringWriter writer = new StringWriter();
        model.write(writer, "TURTLE");
        return writer.toString();
    }}
```

Warto zaznaczyć, że przykładowa aplikacja porusza bardzo obszerne zagadnienie synchronizacji danych. Oczywiste jest, że przesyłanie pełnej informacji o danych z telefonu do systemu NEPOMUK nie może być często powtarzane, gdyż byłoby to wysoce nieefektywne. Należałoby zastosować rozwiązanie oparte na inkrementalnym zbieraniu informacji z telefonu i przesyłaniu do systemu NEPOMUK tylko najnowszych informacji. Można również spróbować stworzyć rozwiązanie oparte na bibliotece Aperture – stworzonej do zbierania informacji z zewnętrznych źródeł danych. System NEPOMUK sam wykorzystuje bibliotekę Aperture do importowania danych z przeglądarek, programu MS Outlook czy Thunderbird.



Rys. 4. Kontakt stworzony w telefonie komórkowym i zaimportowany do systemu NEPOMUK
Fig. 4. Contact created in mobile phone and imported to NEPOMUK system



Rys. 5. Lista zakładek zaimportowana z telefonu komórkowego
Fig. 5. The list of bookmarks imported to mobile phone

4.7. Funkcja pobierania informacji z systemu NEPOMUK

Drugą funkcją jest udostępnienie informacji istniejących w systemie Nepomuk w telefonie komórkowym. Prostym przykładem takiego zastosowania jest poszukiwanie dodatkowych informacji o kontaktach z książki adresowej telefonu. Załóżmy, że użytkownik posiada w książce adresowej telefonu kontakt Claudia Stern (00485000000) oraz tak samo nazwany kontakt w książce adresowej w programie Thunderbird, skojarzony z adresem claudia.stern-

@example.com. Bardzo pożądaną funkcją byłoby zatem wyświetlenie informacji o kontakcie z programu pocztowego bezpośrednio w telefonie.

Podstawowym problemem jest skojarzenie tych dwóch kontaktów. Jeśli program ma pewność, że oba kontakty rzeczywiście odnoszą się do jednej osoby, wystarczyłoby stworzyć trójkę:

```
<semdesk://claudia.stern@example.com/Thing/Claudia> pimo:occurence
<semdesk://00485000000/contacts/1> .
```

Właściwość pimo:occurence oznacza, że podmiot i obiekt odnoszą się do tego samego konceptu. W gramatyce RDF można to przedstawić jako implikacje:

```
(?obiekt ?p ?v) -> (?podmiot ?p ?v)
```

oraz:

```
(?s ?p ?obiekt) -> (?s ?p ?podmiot)
```

Jednakże najpierw należy ustalić, czy istnieje w systemie NEPOMUK odpowiedni kontakt. W tym celu możemy wykorzystać dostępne w telefonie informacje. Jeżeli kontakt w telefonie ma przypisany adres email lub adres strony domowej, możemy skonstruować zapytanie SPARQL do wyszukania kontaktu w systemie NEPOMUK:

```
SELECT  ?x ?y ?z
WHERE { ?x ?y ?z .
    {?x pimo:Occurrence <semdesk://0048506977506/contacts/1> }
    UNION { ?x nco:hasPhoneNumber  <tel:00485977506> }
    UNION { ?x nco:hasEmailAddress <mailto:claudia.stern@example.com> }
    UNION { ?x foaf:homepage <http://www.claudia.stern.example.com> }
    UNION { ?x rdfs:label "Claudia Stern" }
    UNION { ?x nao:prefLabel "Claudia Stern" }}
```

Takie zapytanie zwróci wszystkie trójki, których podmiotem jest:

```
<semdesk://claudia.stern@example.com/Thing/Claudia>.
```

Może się okazać, że aplikacja kliencka nie posiada żadnych informacji poza numerem telefonu i nazwą kontaktu, natomiast system NEPOMUK nie dysponuje numerem telefonu, a nazwę kontaktu stanowi wyłącznie „Claudia”. W takim wypadku nie istnieje możliwość automatycznego skojarzenia obu kontaktów.

Eksperymentalnie zaimplementowane zapytanie SPAQRL, wykorzystujące wyszukiwanie pełnotekstowe, łączy dwie popularne biblioteki Seasame oraz Lucene [22]:

```
PREFIX search:   <http://www.openrdf.org/contrib/lucenesail#>
SELECT DISTINCT ?uri  ?label
WHERE {
    ?uri rdf:type pimo:Person .
    ?uri search:matches ?match.
    ?match search:query "contanctName".
    ?uri rdfs:label ?label . }
```

Posiadając URI kontaktu z systemu NEPOMUK, można jednym zapytaniem pobrać wszystkie informacje o kontakcie. Warto zwrócić w tym momencie uwagę, że prezentacja

danych zależy od aplikacji klienckiej. W większości zastosowań nie są potrzebne wszystkie możliwe informacje o danym zasobie dostępne w sieci semantycznej – może być ich bardzo dużo. Dlatego w niniejszej aplikacji ograniczono zbiór interesujących informacji do następujących:

- adres strony internetowej,
- pliki związane z kontaktem,
- tagi lub notatki związane z kontaktem,
- dodatkowe informacje, np. organizacje lub wydarzenia związane z kontaktem.

W aplikacji mobilnej prezentowane są wybrane informacje, a niektóre zapytania składają się z więcej niż jednej trójki:

```
SELECT ?relatedLabel
WHERE {
    <semdesk://claudia.stern@example.com/Thing/Claudia> pimo:isRelated ?x .
    ?x rdfs:label ?relatedLabel . }
```

Po odczytaniu informacji z repozytorium RDF, tworzone są obiekty POJO (Plain Old Java Object), które są przesyłane są do aplikacji mobilnej.

Aplikację można testować w następujący sposób:

1. W systemie Nepomuk użytkownik tworzy nowy kontakt.
2. Użytkownik wybiera opcję „Show Nepomuk Info”.
3. Użytkownik może użyć kontaktu z książki adresowej lub wpisać dowolną frazę.
4. Jeżeli system Nepomuk posiada więcej niż jeden pasujący wynik (w przypadku omawianym wcześniej), należy wybrać jeden.
5. Wyświetlane są informacje w postaci tekstowej.

5. Podsumowanie

W artykule przedstawiono przykład aplikacji mobilnej wymieniającej dane z aplikacją desktopową za pomocą RDF. Dzięki temu aplikacja jest w stanie wymieniać informacje nie tylko z systemem Nepomuk, ale dowolnym agentem semantycznym. Okazało się, że istnieją już ontologie do reprezentacji dowolnej informacji dostępnej w telefonie. Ontologie stworzone w ramach projektu NEPOMUK pozwalają opisywać pliki, zakładki i inne informacje dostępne zarówno w smartfonie, jak i komputerze desktopowym, natomiast ontologia SIMS jest wystarczająca do opisania specyficznej wiedzy domenowej, dotyczącej rozmów czy wiadomości.

BIBLIOGRAFIA

1. Strona projektu Semantic Desktop, http://www.semanticdesktop.org/.
2. Sauermann L., Bernardi A., Dengel A.: Overview and Outlook on the Semantic Desktop. DFKI, 2005.
3. Strona projektu Nepomuk, http://nepomuk.semanticdesktop.org/.
4. Bergmann O.: Search engine PIM. 2008.
5. Strona projektu Strigi, http://strigi.sourceforge.net/.
6. Woerndl W., Woehrl M.: SeMoDesk: Towards a Mobile Semantic Desktop. Technische Universitaet Muenchen.
7. Junwu Z., Bin L., Fei W., Sicheng W.: Mobile Ontology. JDCTA 4(5), 2010.
8. Lassila O.: Applying Semantic Web in Mobile and Ubiquitous Computing: Will Policy-Awareness Help? Nokia Research Center.
9. Yun H., Su-Kyoung K., YoungTaek J.: A Context-Aware Framework using Ontology for Smart Phone Platform. International Journal of Digital Content Technology and its Applications, Vol. 4, No. 5, 2010.
10. Chen H., Finnin T.: An Ontology for Context Aware Pervasive Computing Environments. Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review, 2003.
11. Klyne G., Reynolds F., Woodrow C. et al.: Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. W3C Recommendation. http://www.w3.org/TR/CCPP-struct-vocab/. 2004.
12. Preuveneers D., Van Den Bergh J., Wagelaar D. et al.: Towards an Extensible Context Ontology for Ambient Intelligence. 2004.
13. Cantera-Fonseca J. M., Lewis R.: Delivery Context Ontology. W3C, 2009.
14. Chen H., Finnin T., Joshi A.: The SOUPA Ontology for Pervasive Computing. In Ontologies for Agents: Theory and Experiences, 2005, s. 233÷258.
15. Poveda-Villalón M., Suárez-Figueroa M. C. et al.: A Context Ontology for Mobile Environments. Proceedings of Workshop on Context, Information and Ontologies, 2010.
16. Domaszewicz J., Cieślak P., Rój M., Rybicki T. Sanders R., Floch J.: Projekt SIMS – Interfejsy Semantyczne w Usługach Mobilnych.
17. Strona Politechniki Warszawskiej, na której opublikowane są ontologie SIMS http://www.tele.pw.edu.pl/~sims-onto/.
18. http://dev.nepomuk.semanticdesktop.org/download/.
19. http://openjena.org.
20. GSM Europe, GSME proposals regarding mobile theft and IMEI security. 2003.

21. Strona projektu LuceneSail, http://www.openrdf.org/contrib/lucenesail.
22. The Sesame LuceneSail: RDF Queries with Full-text Search NEPOMUK Technical Report 2008-1.
23. Mobile and Embedded Applications Group – The Ontology of Telecommunication Services, http://meag.tele.pw.edu.pl/sims/ontology.html.
24. Berners-Lee T.: The Semantic Web – A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. Scientific American Magazine, 2001.
25. Skulimowski M.: Analizator Internetu Semantycznego. Studia Informatica, Vol. 32, No. 2A (96), Wydawnictwo Politechniki Śląskiej, Gliwice 2011.
26. Goczyła K., Piotrowski P.: Application of Knowledge Views. Studia Informatica, Vol. 31, No. 2A (89), Wydawnictwo Politechniki Śląskiej, Gliwice 2010.

Wpłynęło do Redakcji 17 stycznia 2012 r.

Abstract

The Semantic Web standards have already emerged and are widely used, they are mature standards to share knowledge. Semantic Desktop is an initiative which tries to integrate desktop applications with the web, allowing easy data and metadata exchange. Nevertheless there is lack of such exchange between desktop and mobile platform directly. Modern mobile phones provides many features and produce lots of information. There is a need to manage those information and made it easily accessible from desktop programs. This is main concern of knowledge workers. In order to describe information on mobile phone it was necessary to find suitable ontology. Several ontologies were analyzed and presented here. The paper describes example program, which allows to exchange data between Google Android phone and Nepomuk system – PIM application on desktop platform. The paper address problem of unique data identification considering mobile device as a data source. The article tries to find valid RDF representation using existing ontologies for data available on mobile phone. The paper includes several code examples to show:

- Usage of popular RDF library for Java language
- SPARQL queries to retrieve information
- Full-text search extension to SPARQL language

Adresy

Krzysztof CZAJKOWSKI: Politechnika Krakowska, Wydział Fizyki, Matematyki i Informatyki, Instytut Teleinformatyki, ul. Warszawska 24, 31-155 Kraków, Polska, kc@pk.edu.pl.

Tomasz TRELA: Politechnika Krakowska, Wydział Fizyki, Matematyki i Informatyki, Instytut Teleinformatyki, ul. Warszawska 24, 31-155 Kraków, Polska, tomas.trela@gmail.com.