DOI: 10.18668/NG2016.01.08

Jakub Badowski

Instytut Nafty i Gazu – Państwowy Instytut Badawczy

Usługi sieciowe jako narzędzie do nowoczesnego i bezpiecznego udostępniania usług obliczeniowych i badawczych

W artykule omówiono zagadnienie usług sieciowych (ang. *web services*). Przedstawiono różne sposoby implementacji tych usług oraz zasygnalizowano korzyści płynące z ich stosowania. Poruszono temat bezpieczeństwa świadczenia usług w Internecie oraz ochrony własności intelektualnej. Zaprezentowano przykład wykorzystania usługi sieciowej w systemie przeznaczonym do oceny ryzyka eksploatacyjnego gazociągów SOREG®.

Słowa kluczowe: usługi sieciowe, SOREG, SOAP, REST, ochrona własności intelektualnej.

Web services as a tool for innovative and secure sharing of computing and research services

In the article the issue of web services was discussed. Different methods of web services implementation and advantages of their usage were presented. The issue of security of web services in internet and intellectual property protection was discussed. The example of using web services in the risk assessment system of transmission gas pipelines was presented.

Key words: web services, SOREG, SOAP, REST, intellectual property protection.

Wprowadzenie

Wraz z rozwojem Internetu pojawiają się nowe możliwości oferowania rozmaitych usług. Na przestrzeni ostatnich lat powstało wiele firm specjalizujących się w świadczeniu usług sieciowych w zakresie m.in. automatycznej obsługi płatności, dostarczania specyficznych danych, prezentowania map i wielu innych. Okazało się, że znacznie taniej jest skorzystać odpłatnie z gotowego (dobrze sprawdzonego) rozwiązania, niż implementować je na nowo. Powstała nowa koncepcja tworzenia systemów informatycznych – architektura zorientowana na usługi (SOA, ang. *service-oriented architecture*). Upowszechniła się także metoda realizowania usług przez sieć Internet, która przyjęła ogólną nazwę „usługa sieciowa" (ang. *web service*). Artykuł ma na celu przybliżenie czytelnikowi koncepcji usług sieciowych oraz zaprezentowanie możliwości ich wykorzystania do upowszechniania wyników badań oraz nowoczesnego świadczenia usług badawczych.

Czym są usługi sieciowe?

Usługa sieciowa to programistyczna realizacja pewnej funkcjonalności świadczonej (mówiąc w uproszczeniu) za pomocą sieci Internet. Innymi słowy, jest to programistyczna „czarna skrzynka" – program, który wykonuje pewną pracę, a którego zawartość nie ma istotnego znaczenia z punktu widzenia klienta tej usługi. Klienta interesuje jedynie sama funkcjonalność, a dowiaduje się o niej poprzez opis usługi, który powinien zawierać rzetelne informacje na temat jej rodzaju, danych wejściowych, jakich ta usługa oczekuje (tzw. wsadu), oraz danych

wyjściowych (wyniku). Istotny jest fakt, że sposób realizacji funkcjonalności usługi pozostaje ukryty (ukrywanie szczegółów implementacyjnych). Wiele otwartych i komercyjnych serwisów udostępnia tzw. interfejs programistyczny API (ang. *application programming interface*), umożliwiający korzystanie z metod odpowiadających poszczególnym funkcjom serwisu. Dobrym przykładem jest tutaj firma Allegro. Udostępnia ona szereg metod, dzięki którym programista może np. stworzyć niezależną stronę internetową prezentującą ogłoszenia, które wcześniej użytkownik zamieścił w serwisie Allegro.

Częstym błędem jest mylenie usług sieciowych z usługami świadczonymi w sieci Internet takimi jak strony WWW, poczta elektroniczna, komunikatory internetowe itp. Oczywiście strona WWW może być interfejsem służącym do interakcji z użytkownikiem w ramach usługi sieciowej, jednak nie to stanowi istotę sprawy.

W systemach informatycznych, które spełniają warunki architektury SOA (ang. *service-oriented architecture*), usługą sieciową jest każdy element oprogramowania, który posiada niżej określone cechy:

- udostępnia pewną zamkniętą funkcjonalność,
- może działać niezależnie od innych części oprogramowania,
- ma zdefiniowany interfejs do komunikacji,
- działa niezależnie od platformy programistyczno-sprzętowej.

Największą zaletą stosowania usług internetowych jest możliwość korzystania z rozproszonych zasobów (programów udostępniających różne funkcjonalności) niezależnie od języka programowania, w którym są napisane, od platformy sprzętowej, na której działają, itp. Taką koncepcję tworzenia oprogramowania określa się na świecie nazwą *service-oriented architecture* (SOA). Sięganie po tę technikę programistyczną jest zasadne, gdy występuje dobrze zdefiniowany problem (np. autorski algorytm), który można przekształcić w program komputerowy. Z powstającego programu mają w zamierzeniu jego twórców korzystać inni (darmowo lub odpłatnie), jednak szczegóły implementacyjne powinny pozostać ukryte (ochrona własności intelektualnej). Jedną z podstawowych zalet usług sieciowych, które skłoniły autora do przybliżenia tego zagadnienia, jest możliwość świadczenia rozmaitych usług (np. wykorzystywanie autorskiego algorytmu, programu komputerowego czy specyficznych zestawów danych) w skali globalnej przy całkowitej ochronie własności intelektualnej. Jest to możliwe, gdyż komponenty usługowe działają zdalnie, co oznacza, że nie są fizycznie wręczane konsumentom usług.

Należy zwrócić uwagę, w jaki sposób obecnie odbywa się proces np. przeliczania danych za pomocą autorskich programów komputerowych. W wielu przypadkach dane są dostarczane na nośnikach danych (CD, DVD, pamięć USB) lub za pośrednictwem poczty elektronicznej. Twórca programu wprowadza otrzymane dane do programu, uruchamia go i zwraca otrzymane wyniki tą samą drogą. Gdyby jednak klient chciał korzystać z programu bardzo często, taka sytuacja stałaby się kłopotliwa. Zdaniem autora w niektórych, uzasadnionych przypadkach proces ten można w pełni zautomatyzować, co zaoszczędzi czas, z drugiej zaś strony udostępnienie programu w sieci Internet pozwoli znacznie poszerzyć grono potencjalnych zainteresowanych daną usługą.

Badania własne autora wskazują, że w Instytucie Nafty i Gazu – Państwowym Instytucie Badawczym są świadczone usługi oraz wykonywane badania, które z powodzeniem można dystrybuować/popularyzować za pomocą usług internetowych. Dobrym przykładem jest pomysł implementacji usługi, której funkcjonalnością będzie modelowanie pola falowego w ośrodkach anizotropowych. Mógłby to być projekt edukacyjno-reklamowy, kierowany do studentów kierunku geofizyka. Kolejny pomysł to przygotowanie usługi sieciowej, która realizowałaby funkcjonalność prognozowania odzysku metanu z wysypisk komunalnych. Każdy z wymienionych przykładów mógłby z powodzeniem stać się podstawą działania usługi internetowej. W niniejszej publikacji autor zaprezentował przykład wykorzystania usługi sieciowej w systemie przeznaczonym do oceny ryzyka eksploatacyjnego gazociągów.

Rodzaje protokołów

Usługi *web services* (WS) dają możliwość wymiany informacji w postaci dokumentów XML (ang. *Extensible Markup Language*) za pomocą protokołu HTTP [5]. Przykładem wykorzystania WS może być automatyczne pobranie kolekcji informacji w łatwy do przetwarzania sposób. Dane mogą dotyczyć czegokolwiek: kursów walut, rozkładu lotów czy cen akcji na giełdzie. Mimo że dane pochodzą z odległego źródła, dla projektanta aplikacji korzystającej z wyżej wymienionego serwisu operacja ta nie różni się w zasadzie niczym od wywołania lokalnej funkcji.

Wielką zaletą usług WS jest ich niezależność od platformy systemowej czy języka programowania. Przykładowo, skrypt napisany w języku PHP i wykonywany w systemie Debian może bez przeszkód komunikować się z windowsowym odległym serwerem IIS (ang. *Internet Information Services*). Zasadniczo istnieją dwa najbardziej znane protokoły udostępniania

usług sieciowych: REST (ang. *Representational State Transfer*) oraz SOAP (ang. *Simple Object Access Protocol*).

Protokół REST

Protokół REST jest przede wszystkim mało skomplikowany, co stanowi jego największą zaletę. Istota jego działania polega na generowaniu żądań HTTP oraz przetwarzaniu dokumentów XML, które są zwracane w odpowiedzi. Do wad tego mechanizmu możemy zaliczyć fakt, że wysyłanie i pobieranie danych z wykorzystaniem REST nie zostało opisane żadnym standardem. Wynika z tego, że programista danego serwisu ma pełną dowolność w wyborze rozwiązania, a co za tym idzie – każde takie rozwiązanie może być inne, w zależności od preferencji programisty. Nie stanowi to problemu przy małych projektach, przeciwnie – podnosi elastyczność narzędzia oraz ułatwia implementację, co z kolei przekłada się na dużą popularność tego rozwiązania. Niestety, w przypadku bardziej rozbudowanych projektów używanie REST może zwiększać złożoność aplikacji.

Udostępnianie metod REST

Celem utworzenia i udostępnienia metody REST jest umożliwienie zdalnym użytkownikom dostępu do zasobów, które znajdują się na serwerze. Takie rozwiązanie pozwoli użytkownikom na zadawanie sparametryzowanych żądań i otrzymywanie odpowiedzi w postaci dokumentu XML. Najprostszym przypadkiem może być adres zasobu na serwerze WWW, na którym działa skrypt analizujący parametry przesyłane w ciągu zapytania HTTP i zwracający odpowiednio spreparowany kod XML. Przykład skryptu realizującego wyżej wymieniony scenariusz przedstawia rysunek 1.

```php
<?php
// Przykładowe dane
$params_database = <<<_MODEL_
<?xml version="1.0" encoding="utf-8" ?>
<groupA>
 <param id="1">
    <name>System jakości wdrożony u projektanta</name>
    <desc>Parametr logiczny (typu tak/nie)...</desc>
 </param>
 <!-- Miejsce na kolejne parametry -->
 <param id="21">
    <name>Ocena wykonawstwa</name>
    <desc>Ocena na podstawie posiadania systemu jakości...</desc>
 </param>
</groupA>
_MODEL_;

// Załadowanie danych
$s = simplexml_load_string($params_database);

// Wyszukanie danych
$param = addslashes($_GET['param']);

// Zapytanie XPath
$query = "/groupA/param[name = '$param']";

$output = $s->xpath($query);

// Wygenerowanie wyniku wyszukiwania w formie dokumentu XML
header('Content-type: application/xml; charset="utf-8"');
print "<?xml version=\"1.0\" encoding=\"utf-8\" ?>\n";
print "<groupA>\n\t";
foreach ($output as $a) {
    print $a->asXML();
}
print "\n</groupA>";
?>
```

Rys. 1. Prosta usługa typu REST

Zakładając, że powyższa usługa jest dostępna pod adresem *http://api.domena.pl/*, przesłanie do niej żądania *http://api.domena.pl/?param=Ocena+wykonawstwa* spowoduje zwrócenie przez serwer dokumentu XML zawierającego nazwę oraz opis poszukiwanego parametru. W wyżej wymienionym przypadku obsługa żądań REST przypomina zwykłą operację przetwarzania żądania otrzymanego z formularza HTML. Główna różnica polega na tym, że w odpowiedzi generowany jest kod XML. Przekazywanie danych wejściowych realizuje się przez umieszczenie ich w ciągu zapytania URL, a w związku z tym w przypadku języka PHP można do nich dotrzeć w skrypcie, korzystając ze zmiennej tablicowej $_GET. Aby wygenerować właściwą odpowiedź, należy przeanalizować zawartość tej tablicy. Przedstawiony powyżej przykład ma na celu wygenerowanie odpowiedzi w postaci dokumentu XML na zapytanie otrzymane od klienta za pośrednictwem metody GET protokołu HTTP. Na uwagę zasługuje też sposób przeszukiwania dokumentów XML zawarty w skrypcie – XPath. Jest to standard W3C przeznaczony do wyodrębniania określonych informacji z dokumentów XML [15].

Generowanie żądań REST

Za pomocą protokołu REST można generować żądania HTTP, wykorzystując takie metody jak GET, POST, PUT oraz DELETE. Używana metoda stanowi jednocześnie informację dla serwera o tym, jakie działanie należy podjąć. Dla przykładu metoda GET oznacza chęć pobrania danych ze zdalnego serwera, natomiast metoda POST sygnalizuje

zamiar przesłania pewnych informacji od klienta na serwer. Aplikacja serwera odsyła wynik zapytania w formie dokumentu XML. Dokument ten można już samodzielnie poddać dalszemu przetwarzaniu.

Prostota mechanizmu REST stanowi jego największą zaletę. Wykorzystuje on istniejące od lat standardy, zatem wszystkie funkcje potrzebne do generowania i analizowania żądań REST są już zaimplementowane i gotowe do użycia.

W języku PHP żądanie HTTP może być wygenerowane na kilka sposobów, np. z wykorzystaniem:

- funkcji *file_get_contents()* [8],
- rozszerzenia cURL [1],
- pakietu PEAR [7].

W języku PHP istnieje wiele rozszerzeń do analizy pobranego dokumentu XML. W przypadku nieskomplikowanych struktur dokumentów najlepiej wykorzystać rozszerzenie SimpleXML, natomiast do analizy dokumentów bardziej rozbudowanych warto zastanowić się nad użyciem takich narzędzi jak XMLReader czy uniwersalny język przekształcania dokumentów XML – XSLT [6].

Protokół SOAP

Drugim najbardziej powszechnym sposobem udostępniania usług *web services* jest protokół SOAP (ang. *Simple Object Access Protocol*). Według najogólniejszej definicji jest to „protokół wywoływania zdalnego dostępu do obiektów, wykorzystujący XML do kodowania wywołań i najczęściej protokoły HTTP lub RPC do ich przenoszenia, możliwe jest jednak wykorzystanie innych protokołów do transportu danych" [10]. Definiuje go standard W3C [14]. W standardzie tym opisano sposób przesyłania komunikatów przez sieć oraz mechanizm wywoływania funkcji na odległych maszynach. Dzięki rozwiązaniom SOAP programiści otrzymują bardzo potężne narzędzie, o rozbudowanych możliwościach. Z drugiej strony wielkość SOAP powoduje komplikację implementacji protokołu, chociaż paradoksalnie pierwotnym celem tego mechanizmu była próba ułatwienia projektowania aplikacji.

Komunikacja między usługami odbywa się w architekturze klient–serwer. Obie strony komunikacji automatycznie „serializują" typy danych. Możliwe jest więc przekazywanie oraz pobieranie złożonych danych bez obawy o wystąpienie problemów z ich zgodnością. Dzięki temu programista ma pewność, że wymiana informacji odbywa się bez problemów. Uciążliwym niuansem protokołu SOAP jest trudność wykrywania ewentualnych błędów w działaniu serwisu w przypadku braku dogłębnej znajomości języka XML wykorzystywanego w niższych warstwach. Powodem tego jest fakt, że warstwa SOAP działa ponad warstwą komunikacyjną [11].

Implementacja SOAP w języku PHP

SOAP jest z założenia protokołem wymiany informacji niezależnym od platformy. Dzięki temu doczekał się implementacji w wielu językach programowania (np. Java, C, C++, C#, Python). Wszystkie przykłady kodu zawarte w niniejszym artykule (zarówno od strony klienta, jak i serwera) są napisane w języku PHP.

Interpreter PHP został wyposażony w kilka implementacji SOAP, np. PEAR::SOAP czy NuSOAP. Od wersji PHP 5 zaleca się korzystanie z wbudowanego rozszerzenia, ponieważ jest ono dostarczane wraz z interpreterem i domyślnie włączone, jest szybkie (napisane w języku C), zgodne ze specyfikacją SOAP oraz obsługuje mechanizm wyjątków.

Udostępnianie metod w protokole SOAP

SOAP daje możliwość utworzenia usługi serwerowej, która będzie odpowiadała na żądania klientów. Do realizacji tego zadania wykorzystano rozszerzenie ext/soap SOAPServer. Przykładowy kod prostego serwera usługi zaprezentowano na rysunku 2.

Procedura uruchamiania serwera składa się z trzech etapów:

1) napisanie klasy, która będzie obsługiwała metody SOAP,
2) utworzenie obiektu serwera i skojarzenie z nim klasy obsługującej metody SOAP,
3) uruchomienie funkcji przetwarzania żądań i odsyłania odpowiedzi w serwerze SOAP.

W przedstawionym przykładzie w liniach 23–27 widać klasę *pc_SOAP_return_model*, zawierającą jedną metodę o nazwie *return_model*. Klasa ta jest wykorzystywana do obsługi żądań SOAP. Kolejny krok po zdefiniowaniu klasy to powołanie obiektu typu *SOAPServer* (linia 29). Gdyby usłudze towarzyszył plik WSDL, należałoby przekazać jego nazwę jako parametr konstruktora obiektu. Prezentowany przypadek nie korzysta z dokumentu WSDL, więc w miejscu pierwszego parametru należy wpisać wartość *null*. Drugi parametr konstruktora przeznaczony jest do wprowadzania opcji konfiguracyjnych. Wymagane minimum to jedno ustawienie, które określa przestrzeń nazw serwera SOAP (*'uri' => 'urn:pc_SOAP_return_model'*). Do tego momentu nazwa klasy PHP nie ma szczególnego znaczenia – ważna jest przestrzeń XML, określona w tym przypadku jako *urn:pc_SOAP_return_model*.

```php
<?php

class Model
{
    public $version = "2.0";
    public $author = "Jakub Badowski";
    public $desc = "Model punktowy systemu SOREG";

    public $params = array (
                                array ( "number" => "1",
                                        "name" => "System jakości wdrożony u projektanta",
                                        "value" => "2"
                                        ),

                                array ( "number" => "1",
                                        "name" => "System jakości wdrożony u operatora",
                                        "value" => "5"
                                        )
                                        // Reszta parametrów
                                );
}

class pc_SOAP_return_model {
    public function return_model() {
        return new Model();
    }
}

$server = new SOAPServer(null, array('uri' => 'urn:pc_SOAP_return_model'));
$server->setClass('pc_SOAP_return_model');
$server->handle();
```

Rys. 2. Prosty serwer usługi SOAP

Kolejnym krokiem, widocznym w linii 30, jest wywołanie metody *SOAPServer::setClass()*. Pierwszy i w tym przypadku jedyny argument tej metody stanowi nazwa klasy. Metody tej klasy będzie próbował wywołać serwer SOAP w momencie odbierania żądań. Ostatnim krokiem jest wydanie rozkazu uruchomienia serwera, który będzie odpowiadał na żądania klientów. Służy do tego metoda *SOAPServer::handle()*. Kojarzenie całej klasy z obiektem SOAPServer to najbardziej popularne rozwiązanie. Niemniej jednak możliwe jest również wiązanie z obiektem serwera usług pojedynczych funkcji.

Wywołanie metody SOAP

Powyżej został omówiony sposób udostępniania zdalnym klientom metod za pomocą protokołu SOAP. Aby klient mógł z nich korzystać, musi mieć możliwość wywołania wyżej wymienionych metod w swoim programie. Może to zrobić za pomocą protokołu SOAP poprzez wysyłanie żądań, używając do tego celu klienckiej aplikacji napisanej w dowolnym języku programowania. Na rysunku 3 zaprezentowano przykład napisany w języku PHP.

W celu wygenerowania żądania SOAP należy utworzyć obiekt klasy SOAPClient. W zależności od tego, czy usługa będzie korzystać z pliku WSDL, czy nie, jako parametr przekazywany jest odpowiednio adres określający położenie pliku WSDL lub wartość *null*. Zaprezentowany przykład nie korzysta z danych WSDL, więc konieczne jest ręczne określenie parametrów. O dokumentach oraz samym języku opisu usług WSDL autor wspomina w dalszej części artykułu. Parametry usługi, tj. lokalizacja (adres) oraz ciąg URI przestrzeni nazw, są przekazywane do konstruktora obiektu jako tablica asocjacyjna. Tablicę parametrów widać na rysunku 3 w liniach 2 i 3. Do przesłania żądania wykorzystuje się metodę *__soapCall*. W parametrach przekazywana jest nazwa metody zdalnej oraz tablica parametrów tej metody. W przypadku braku parametrów będzie to pusta tablica. Przesyłanie parametrów w formie tablicy jest podyktowane tym, że obiekt klienta SOAP nie posiada informacji o liczbie wymaganych parametrów. Sytuacja wygląda prościej w przypadku wykorzystującym dokument WSDL, gdy zdalne metody wywoływane są bezpośrednio na obiekcie SOAP.

Protokół SOAP daje możliwość pobierania parametrów żądania, na podstawie których serwer generuje właściwą odpowiedź. Odczyt parametrów wymaga wprowadzenia modyfikacji w prototypach metod, w których należy uwzględnić nazwy tych parametrów. Żądania mogą wówczas zawierać dodatkowe informacje. Przykład wywołania zdalnej metody z parametrami prezentuje skrypt widoczny na rysunku 4.

```php
<?php
$opts = array('location' => 'http://services.lamp1.inig.pl/model.php',
              'uri' => 'urn:pc_SOAP_return_model');

try {
    $client = new SOAPClient(null, $opts);
    $model = $client->__soapCall('return_model', array());

} catch (SoapFault $e) {
    print_r($client);
}

echo '<pre>';
print_r($model);


?>
```

Rys. 3. Przykład wywołania metody SOAP w języku PHP

```php
<?php
class pc_SOAP_return_model {
    public function return_model($rodzaj)
    {
        // Zwraca parametry związane z wysokim ciśnieniem
        if ($rodzaj == 'WC')
        {
            $data = show_model('WC');
        }
        // Zwraca parametry związane z niskim ciśnieniem
        elseif (condition) {
            $data = show_model('NC');
        }
        // Zwraca wszystkie parametry
        else
        {
            $data = show_model();
        }
        return $date;
    }
}

$server = new SOAPServer(null,array('uri'=>'urn:pc_SOAP_return_model'));
$server->setClass('pc_SOAP_return_model');
$server->handle();
?>
```

Rys. 4. Parametry w metodach SOAP

Uwierzytelnianie

W celu zabezpieczenia usługi hasłem, tak aby była ona dostępna tylko dla wybranych użytkowników/klientów, można zastosować mechanizm uwierzytelniania. Najpopularniejszą metodą uwierzytelniania usług sieciowych jest wykorzystanie mechanizmu HTTP Basic Authentication [4]. Cała operacja sprowadza się do umieszczenia odpowiednich danych w nagłówku SOAP zgodnie ze specyfikacją *WS-Security*. Aby wymusić dodawanie danych uwierzytelniających HTTP Basic Authentication do każdego żądania przy tworzeniu obiektu klienta SOAP, należy w parametrze z opcjami przekazać login oraz hasło użytkownika/klienta. Przykład implementacji takiego klienta został zaprezentowany na rysunku 5.

Język opisu usług WSDL

WSDL (ang. *Web Services Description Language*) [13] jest językiem opisu usług *web services*. Tworzy się w nim dokumenty XML zawierające definicje metod oraz parametrów obsługiwanych przez projektowaną usługę. Plik przygotowany zgodnie ze standardem WSDL należy umieścić na serwerze w celu udostępnienia go klientom usługi. Dokument w formacie WSDL nie wygląda zbyt przyjaźnie dla człowieka i nie jest łatwym przedmiotem analizy, jednak

```php
<?php
$opts = array('location' => 'http://services.lamp1.inig.pl/model.php',
              'uri' => 'urn:pc_SOAP_return_model',
              'login' => 'user',
              'password' => '$djs24kf0b24#');

try {
    $client = new SOAPClient(null, $opts);

} catch (SoapFault $e) {
    print_r($client);
}

echo '<pre>';
print_r($client->return_model(false));

?>
```

Rys. 5. Uwierzytelnianie usług w SOAP

stanowi bezcenne źródło informacji o usłudze dla komputerów. Wskazanie położenia dokumentu WSDL sprawia, że mechanizm SOAP automatycznie utworzy obiekt potrzebny do komunikacji z daną usługą oraz umożliwi programiście używanie go w sposób analogiczny do lokalnych obiektów innych klas PHP. We wspomnianym obiekcie zapisane są informacje o parametrach pobieranych przez metody danej usługi oraz o ich typach. Jest to szczególnie istotne ze względu na fakt, że język SOAP uwzględnia ścisłą kontrolę typów. Nie można na przykład przekazać w parametrze metody ciągu tekstowego '5', podczas gdy spodziewaną wartością jest liczba 5.

Realizacja przykładowej usługi

Jako przykład realizacji usługi sieciowej autor wybrał zagadnienie związane z autorskim Systemem Oceny Ryzyka Eksploatacyjnego Gazociągów (SOREG®). System ten powstał w Instytucie Nafty i Gazu – Państwowym Instytucie Badawczym (INiG – PIB) i działa jako aplikacja internetowa [2, 3]. W trakcie realizacji przedsięwzięcia pojawiło się kilka problemów, które z powodzeniem można by rozwiązać, wykorzystując techniki związane z usługami sieciowymi.

Pierwszym problemem okazała się sprawa upublicznienia modelu punktowego, stanowiącego podstawę systemu eksperckiego i będącego własnością intelektualną ekspertów, którzy go opracowali. Powstaje zatem pytanie, jak pogodzić możliwość sprzedaży systemu (z czym wiąże się konieczność przekazania kodu źródłowego) z potrzebą ochrony własności intelektualnej. Można tego dokonać, implementując sposób oceny odcinka gazociągu zgodnie z modelem punktowym jako metodę działającą na zdalnym serwerze (w tym przypadku należącym do INiG – PIB). Taka metoda byłaby dostępna tylko dla użytkowników (klientów) uwierzytelnionych, czyli posiadających takie dane jak adres serwera, na którym znajduje się usługa, login oraz hasło. W tym przypadku klientem usługi byłaby aplikacja sprzedawana do firmy zewnętrznej jako produkt.

Przygotowanie środowiska serwerowego

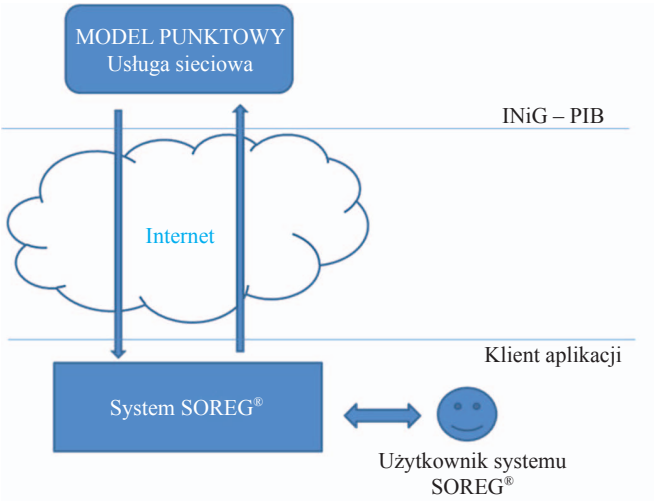
Aplikacja prezentowana w niniejszej pracy łączy się ze zdalnym komputerem (serwerem) za pośrednictwem protokołów komunikacyjnych oraz sieci Internet. W tym celu autor przygotował odpowiedni serwer oraz skonfigurował na nim niezbędne usługi (programy komputerowe działające w tle). Oto lista najważniejszych czynności, które należało wykonać:

1. Stworzenie maszyny wirtualnej w środowisku wirtualizacyjnym VMware [9].
2. Instalacja i konfiguracja systemu operacyjnego Ubuntu 12.04.2 LTS (GNU/Linux 3.5.0-23-generic x86_64) [12].
3. Konfiguracja serwera bazy danych MySQL.
4. Instalacja systemu SOREG®.
5. Instalacja usługi sieciowej (*web service*).

Integracja systemu SOREG® z usługą sieciową

System wspomagania decyzji SOREG® opiera się na modelu punktowym, zaprojektowanym przez specjalistów z INiG – PIB. Należy znaleźć takie rozwiązanie, aby klienci zainteresowani kupnem systemu, mimo otrzymania kodu źródłowego aplikacji, nie mieli dostępu do modelu punktowego, będącego własnością intelektualną jego twórców, chociaż oceny gazociągów mają być dokonywane według tego właśnie modelu punktowego. Problem można rozwiązać, przygotowując usługę sieciową, która przechowuje szczegóły modelu punktowego i pozwala oceniać gazociągi z wykorzystaniem tego modelu za pomocą specjalnego interfejsu programistycznego. Schemat działania byłby następujący: system SOREG®, wywołując odpowiednią metodę usługi

sieciowej, pobiera aktualną listę parametrów do oceny. Użytkownik systemu SOREG® dokonuje oceny danego gazociągu zgodnie z tą listą, nie widząc wag liczbowych poszczególnych parametrów. Po zatwierdzeniu oceny wywołana zostaje kolejna metoda, która w parametrze przekazuje tablicę ocenionych parametrów. Odpowiednia metoda zdalnej usługi dokonuje obliczeń i klasyfikuje dany odcinek, przypisując go do jednej z pięciu kategorii ryzyka. Informacja o tym, która kategoria została przypisana do danego odcinka gazociągu, trafia do aplikacji klienckiej jako finalny wynik działania usługi sieciowej. Uproszczony schemat wykorzystania takiej usługi zaprezentowano na rysunku 6.



Rys. 6. Schemat implementacji modelu punktowego jako usługi sieciowej

Podsumowanie

Zaprezentowany przez autora przegląd możliwości wykorzystania usług sieciowych powinien być wskazówką i zachętą do szerszego stosowania tej techniki w procesie upowszechniania wyników badań oraz nowoczesnego świadczenia usług badawczych. Zdaniem autora takie podejście może zmienić na lepsze dotychczasowe spojrzenie na współpracę pomiędzy nauką a przemysłem. Zakres wykorzystania niniejszej pracy może być bardzo szeroki. Odpowiednio zaimplementowane usługi sieciowe można z powodzeniem zastosować w praktycznie każdej dziedzinie nauki. Konkretne wdrożenia niniejszej pracy mogą pozytywnie wpłynąć na ilość i jakość wykorzystywania prac naukowo-badawczych w przemyśle.

Prosimy cytować jako: Nafta-Gaz 2016, nr 1, s. 58–65, DOI: 10.18668/NG2016.01.08

Artykuł nadesłano do Redakcji 10.09.2015 r. Zatwierdzono do druku 16.10.2015 r.

Artykuł powstał na podstawie pracy statutowej pt. *Wykorzystanie usług sieciowych (Web Services) do upowszechniania wyników badawczo-naukowych* – praca INiG – PIB na zlecenie MNiSW; nr archiwalny: DK-5100-52/14, nr zlecenia: 0078/14/01.

Literatura

[1] cURL. http://curl.haxx.se/ (dostęp: 8.09.2015).
[2] Dietrich A.: *Gas pipeline risk assessment by Internet application*. Nafta-Gaz 2012, nr 11, s. 817–820.
[3] Dietrich A., Badowski J.: *System komputerowy oceny stanu technicznego i analizy ryzyka dla dystrybucyjnych sieci gazowych*. Nafta-Gaz 2009, nr 11, s. 895–900.
[4] *HTTP Authentication: Basic and Digest Access Authentication*. Network Working Group. Request for Comments: 2617. http://tools.ietf.org/html/rfc2617 (dostęp: 8.09.2015).
[5] *HTTP – Hypertext Transfer Protocol*. World Wide Web Consortium (W3C). http://www.w3.org/Protocols/ (dostęp: 8.09.2015).
[6] Otegem M.: *XSLT. Uniwersalny język przekształcania dokumentów XML. Dla każdego*. Gliwice, Wydawnictwo Helion, 2003.
[7] *PEAR – PHP Extension and Application Repository*. The PHP Group. https://pear.php.net/ (dostęp: 8.09.2015).
[8] *PHP Manual. Function Reference: file_get_contents*. The PHP Group. http://php.net/manual/en/function.file-get-contents.php (dostęp: 8.09.2015).
[9] *Server Virtualization with VMware vSphere*. VMware, Inc. http://www.vmware.com/products/vsphere/ (dostęp: 8.09.2015).
[10] *SOAP Specifications*. World Wide Web Consortium (W3C). http://www.w3.org/TR/soap/ (dostęp: 8.09.2015).
[11] Trachtenberg A., Sklar D.: *PHP. Receptury*. Gliwice, Wydawnictwo Helion, 2007.
[12] *Ubuntu Server – for scale out computing*. Canonical Ltd. http://www.ubuntu.com/server (dostęp: 8.09.2015).
[13] *Web Services Description Language (WSDL) 1.1*. World Wide Web Consortium (W3C). http://www.w3.org/TR/wsdl (dostęp: 8.09.2015).
[14] World Wide Web Consortium (W3C). http://www.w3.org/ (dostęp: 8.09.2015).
[15] *XML Path Language (XPath). Version 1.0*. World Wide Web Consortium (W3C). http://www.w3.org/TR/xpath/ (dostęp: 8.09.2015).

Mgr inż. Jakub BADOWSKI
Asystent w Zakładzie Informatyki.
Instytut Nafty i Gazu – Państwowy Instytut Badawczy
ul. Lubicz 25 A
31-503 Kraków
E-mail: *jakub.badowski@inig.pl*