STUDIA INFORMATICA 2003
Volume 24 Number 3 (55)

Sławomir KULIKÓW
Politechnika Śląska, Instytut Informatyki

IMPLEMENTACJA SERWERA ANALIZY LINGWISTYCZNEJ DLA SYSTEMU THETOS – TRANSLATORA TEKSTU NA JĘZYK MIGOWY[1]

Streszczenie. W artykule przedstawiono serwer analizy lingwistycznej, z którego korzysta system translacji tekstu pisanego na język migowy - Thetos. Omówiono architekturę systemu oraz podano sposób korzystania z serwera z poziomu innych aplikacji.

Słowa kluczowe: serwer analizy lingwistycznej.

IMPLEMENTATION OF LINGUISTIC ANALYSIS SERVER FOR THETOS – POLISH TEXT INTO SIGN LANGUAGE TRANSLATOR

Summary. Linguistic analysis server is presented in this paper. It is used by Thetos - text into sign language automatic translator for Polish. System architecture is also presented and instructions how to use the server are given.

Keywords: linguistic analysis server.

1. Wprowadzenie

Jesteśmy społeczeństwem informacyjnym, które pragnie mieć łatwy i szybki dostęp do informacji z każdego miejsca na świecie. Ten cel jest coraz bliższy za sprawą zwiększającego się zasięgu sieci Internet. Przy coraz większych rozmiarach danych pojawia się problem z efektywnym udostępnianiem tych danych dla użytkowników. Oprócz samego udostępniania

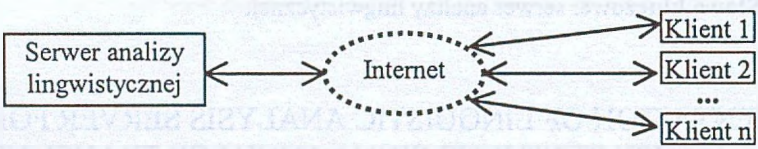
[1] Praca naukowa finansowana ze środków Komitetu Badań Naukowych w latach 2003-2005 jako projekt badawczy 4 T11C 024 24.

użytkownicy coraz częściej żądają wykonania pewnych operacji na tych danych, dlatego należy postawić nacisk na produkcję wydajnych narzędzi do przetwarzania informacji.

Jednym z rodzajów przetwarzania informacji jest przetwarzanie języka naturalnego. W Instytucie Informatyki Politechniki Śląskiej w Gliwicach opracowano komponenty do analizy tekstu dla systemu Thetos [1, 2]. Analiza tekstu może być także wykorzystywana przez inne aplikacje, dlatego uznano za konieczne, żeby część odpowiedzialną za przetwarzanie języka naturalnego wyodrębnić i uczynić z niej serwer analizy lingwistycznej.

2. Architektura systemu

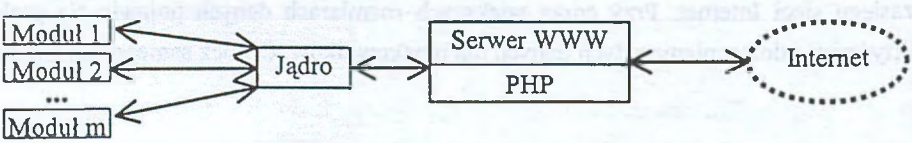
System składa się z serwera analizy lingwistycznej oraz z jednego lub wielu klientów (rys. 1). Takim klientem, oprócz programu Thetos, może być dowolna aplikacja korzystająca z wyników analizy lingwistycznej. Połączenie jest realizowane poprzez sieć Internet za pomocą protokołu HTTP [3]. Cała komunikacja sprowadza się do wysłania żądania klienta do serwera i odbioru odpowiedzi. Przesyłane dane są zapisane w standardzie XML [4].



Rys. 1. Architektura systemu
Fig. 1. System architecture

2.1. Budowa serwera analizy lingwistycznej

Na rysunku 2 przedstawiono serwer analizy lingwistycznej. Składa się on z dwóch komponentów pracujących niezależnie od siebie, a cała komunikacja między nimi odbywa się przez przesył komunikatów. Pierwszym komponentem jest dowolny serwer WWW (np. Apache lub IIS) z obsługą skryptów PHP [5]. Służy on do odbierania żądań i wysyłania wyników. Drugim komponentem jest jądro systemu razem z modułami. Właściwa analiza lingwistyczna jest wykonywana przez poszczególne moduły (w każdym wykonywany jest pewien etap analizy).



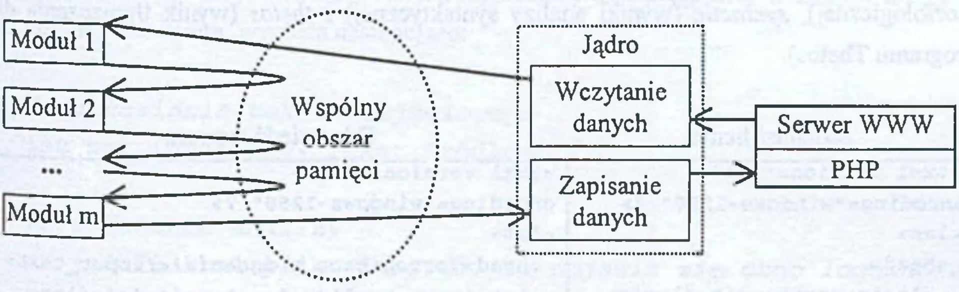
Rys. 2. Budowa serwera
Fig. 2. Server structure

2.1.1. *Budowa modułu*

Moduł jest standardową biblioteką *.dll [6], w której zaimplementowano wspólny interfejs dla wszystkich modułów. Biblioteka powinna być napisana w taki sposób, żeby można było bezpiecznie wywoływać jej funkcje z wielu wątków (ang. *thread safe*). Przy uruchamianiu i zamykaniu serwera wywoływane są funkcje: *las_dll_init* (można przekazać parametry) i *las_dll_done*. Dla każdego żądania klienta tworzony jest nowy wątek, w którym wywoływane są funkcje: *las_init* (można także przekazać parametry), *las_run* (wykonanie właściwej analizy) i *las_done*.

2.1.2. *Budowa jądra*

Zadaniem jądra jest komunikowanie się z serwerem WWW oraz sterowanie pracą modułów. Na rysunku 3 przedstawiono przepływ danych w procesie analizy. Żądanie klienta, które zostało zweryfikowane na poziomie serwera WWW, jest wczytywane do pamięci do struktury drzewa. Następnie w osobnym wątku zgodnie z wybranym łańcuchem przetwarzającym są sekwencyjnie wywoływane moduły, które uzupełniają żądanie o wyniki analizy. Moduły oraz jądro systemu komunikują się ze sobą przez wspólny obszar pamięci. Na końcu analizy na podstawie zmienionego żądania generowana jest odpowiedź w standardzie XML. Dodatkowo przy wczytywaniu żądania następuje konwersja polskich znaków do standardu *windows-1250*, a przy generowaniu odpowiedzi – do standardu źródłowego (np. można wysyłać żądania z polskimi znakami w standardzie *ISO-8859-2*).



Rys. 3. Przepływ danych
Fig. 3. Data flow

2.2. Protokół komunikacyjny

Zastosowany protokół jest oparty na protokole HTTP [3]. Podobnie jak HTTP jest to protokół bezpołączeniowy, gdzie po wysłaniu żądania klienta i odebraniu odpowiedzi następuje zamknięcie połączenia. Dane są przesyłane w postaci tekstu zaraz za nagłówkiem HTTP.

Dostęp do serwera uzyskuje się na podstawie nazwy użytkownika oraz hasła przesyłanego z każdym żądaniem. W odpowiedzi serwera znajduje się żądanie źródłowe (zmodyfikowane – bez hasła) uzupełnione o dodatkowe dane (wyniki analizy, ostrzeżenia, itp.). Na rysunku 4 podano przykładowe żądanie podania nazw łańcuchów oraz odpowiedź serwera.

Żądanie klienta

```
<?xml version="1.0"
encoding="windows-1250" ?>
<las>
  <head>
    <login>uzytkownik</login>
    <password>haslo</password>
  </head>
</las>
```

Odpowiedź serwera

```
<?xml version="1.0"
encoding="windows-1250" ?>
<las>
  <chains>
    <chain>DzielNaSlowa</chain>
    <chain>Morfologia</chain>
    <chain>Syntaktyka</chain>
  </chains>
</las>
```

Rys. 4. Żądanie podania nazw łańcuchów i odpowiedź serwera
Fig. 4. A request of giving chains and a server response

Na rysunku 5 podano przykładowe żądanie wykonania analizy. Jest to rozszerzenie powyższego żądania o nazwę łańcucha przetwarzającego i tekst wejściowy. Przykładowa odpowiedź na żądanie dla łańcucha *Syntaktyka* zawiera nowe sekcje: *log* (czas analizy i ostrzeżenia), *words* (tekst źródłowy podzielony na słowa oraz wyniki analizy morfologicznej), *syntactic* (wyniki analizy syntaktycznej) i *thetos* (wynik tłumaczenia dla programu Thetos).

Żądanie klienta

```
<?xml version="1.0"
encoding="windows-1250" ?>
<las>
 <head>
   <login>uzytkownik</login>
   <password>haslo</password>
   <chain>Syntaktyka</chain>
 </head>
 <input_text>Tekst
wejściowy.</input_text>
</las>
```

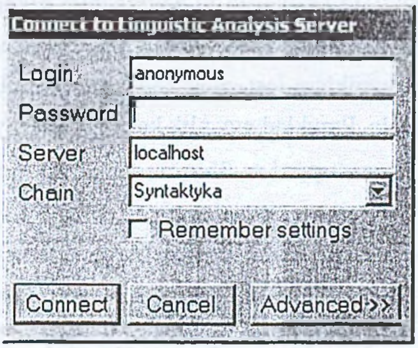
Odpowiedź serwera

```
<?xml version="1.0"
encoding="windows-1250" ?>
<las>
 <head>(przepisane z żądania)</input_text>
 <log>(czas analizy i ostrzeżenia)</log>
 <words>(wyniki analizy morfologicznej)
 </words>
 <syntactic>(wyniki analizy syntaktycznej)
 </syntactic>
 <thetos>(wynik tłumaczenia dla programu
Thetos)</thetos>
</las>
```

Rys. 5. Żądanie wykonania analizy i odpowiedź serwera
Fig. 5. A linguistic analysis request and a server response

2.3. Aplikacja klienta

Aplikacja klienta może łączyć się z serwerem analizy lingwistycznej za pomocą protokołu HTTP (jedyne rozwiązanie dla systemów niekompatybilnych z Microsoft Windows) lub może korzystać z gotowej biblioteki, która ukrywa całą komunikację. W przypadku korzystania z biblioteki konfiguracja połączenia może być automatycznie wczytana z pliku *las.ini*, jak również można poprosić użytkownika o wprowadzenie parametrów (rys. 6).



Rys. 6. Ustawienie parametrów połączenia
Fig. 6. Setup of connection parameters

W bibliotece zaimplementowano funkcje, które ułatwiają odczytanie wyników (pobierają dane ze znaczników: *syntactic* i *thetos*). Przykładowy kod programu, w którym następuje połączenie z serwerem, wygląda następująco:

```
// ustawienie tekstu wejściowego
lac_set_inputtext("Tekst źródłowy");

// wykonanie analizy
// jeżeli brak pliku las.ini, to pojawia się okno logowania
lac_execute();

// odczytanie wyników
printf("Odpowiedź XML: %s\n", lac_get_xml());
printf("Syntaktyka: %s\n", lac_get_syntactic());
printf("Thetos: %s\n", lac_get_thetos());
```

3. Sterowanie procesem analizy

Analiza przebiega zgodnie z danymi zawartymi w pliku konfiguracyjnym. Plik jest zapisany w standardzie XML [4] i składa się z trzech sekcji. Pierwszą sekcją jest opis modułów (*modules*) wykorzystywanych w serwerze analizy lingwistycznej. Dla każdego modułu jest podany jego identyfikator, nazwa pliku *.dll oraz parametry startowe. Następną sekcją jest opis łańcuchów (*chains*). Dla każdego łańcucha jest podana jego nazwa oraz lista kolejno wywoływanych modułów opisanych w poprzedniej sekcji. Ostatnią sekcją jest opis użytkowników (*users*). Dla każdego użytkownika jest podana jego nazwa oraz hasło. Dodatkowo z każdym użytkownikiem jest związana lista łańcuchów, do których ma dostęp po podaniu prawidłowego hasła. Przykładowy plik konfiguracyjny wygląda następująco:

```
<config>

  <modules>
    <module id="ToWords" path="demo\dziel_na_slowa"
            name="dziel_na_slowa.dll" />
    <module id="Morph" path="demo\morph"
            name="analiza_morfologiczna.dll" param="morf" />
    <module id="Syntactic" path="demo\skladnia"
            name="analiza_syntaktyczna.dll" />
  </modules>

  <chains>
    <chain name="Morfologia">
      <module id="ToWords" />
      <module id="Morph" />
    </chain>
    <chain name="Syntaktyka">
      <module id="ToWords" />
      <module id="Morph" />
      <module id="Syntactic" />
    </chain>
  </chains>

  <users>
    <user login="uzytkownik" password="haslo">
      <chain>Morfologia</chain>
```

```
        <chain>Syntaktyka</chain>
      </user>
    </users>
  </config>
```


4. Plany na przyszłość

Przedstawiony w tej pracy serwer analizy lingwistycznej jest dopiero we wczesnej fazie rozwoju i wymaga dopracowania. Należałoby zapewnić szyfrowanie przesyłanego hasła (można do tego celu wykorzystać protokół HTTPS) oraz wprowadzić pewne restrykcje dotyczące użytkowników (np. zakres adresów IP, z których mogą nawiązywać połączenie). Planowana jest adaptacja programów wykonujących analizę jako modułów serwera oraz adaptacja istniejących programów wymagających analizy lingwistycznej do korzystania z serwera (np. generator streszczenia tekstu [7]). Ponieważ moduły do analizy mogą być rozwijane przez różne osoby, dlatego powinna być możliwość zdalnej aktualizacji serwera.

5. Podsumowanie

W niniejszej pracy został zaprezentowany serwer analizy lingwistycznej, z którego mogą korzystać różne aplikacje (jedną z nich jest Thetos). Wprowadzenie architektury klient-serwer pozwoliło uzyskać pewne korzyści. Wśród nich znajdują się ochrona praw autorskich (baza danych oraz aplikacja serwera nie jest bezpośrednio dostępna dla użytkownika) oraz udostępnienie zaawansowanych usług szerszemu gronu odbiorców.

LITERATURA

1. Lubiński M.: Model środowiska systemu przetwarzania danych – translacji języków naturalnych. Studia Informatica, Vol 21, No 3 (41), Gliwice 2000.
2. Suszczańska N., Szmal P., Francik J.: Translating Polish Texts into Sign Language in the TGT System. 20th IASTED International Conference APPLIED INFORMATICS - AI'2002. Innsbruck, Austria 2002. pp. 282-287.
3. Wong C.: HTTP Leksykon kieszonkowy. Helion, Gliwice 2000.
4. North S.: XML dla każdego. Helion, Gliwice 2000.

5. Hughes S.: PHP4. Podręcznik programisty. Helion, Gliwice 2002.
6. Klein M.: Przewodnik po bibliotekach DLL i sposobach zarządzania pamięcią. Intersoftland, Warszawa 1994.
7. Suszczańska N., Kulików S.: A Polish Document Summarizer. 21st IASTED International Conference APPLIED INFORMATICS - AI'2003. Innsbruck, Austria 2003. pp. 369-374.

Recenzent: Dr inż. Ryszard Winiarczyk

Wpłynęło do Redakcji 24 kwietnia 2003 r.

Abstract

A linguistic analysis server, which may be used by various clients (e.g. Thetos application [2] - text into sign language automatic translator for Polish), is presented in this paper. System architecture is presented in section 2. A server structure is shown in fig. 2. The server consists of any web server (for communication between the server and the clients), the kernel (which controls the analysis process) and modules (which realize the analysis). Data flow is shown in fig. 3. Requests from clients and server responses are presented in fig. 4 and 5. A client application may use a dedicated DLL library to connect to the server. Setup window from the library is shown in fig. 6. Section 3 describes a configuration file, which is used to control an analysis process and entity authentication. Directions of future server development are discussed in the ending of this paper.

Adres

Sławomir KULIKÓW: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, skulikow@star.iinf.polsl.gliwice.pl .