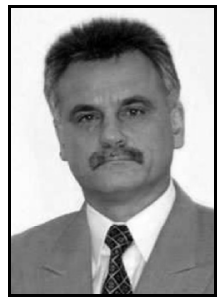
Zdzisław KOWALCZUK, Jan KLIMCZAK
KATEDRA SYSTEMÓW DECYZYJNYCH, WYDZIAŁ ELEKTRONIKI, TELEKOMUNIKACJI I INFORMATYKI, POLITECHNIKA GDAŃSKA,
Narutowicza 11, 80-233 Gdańsk

System inteligentnej nawigacji sterowanej głosem po serwisie internetowym

Prof. dr hab. inż. Zdzisław KOWALCZUK



Prof. dr hab. inż. (2003, 1993, 1986, 1978). Jest profesorem zwyczajnym Automatyki i Robotyki i kier. Katedry Systemów Decyzyjnych. Wizyty: Oulu Univ. (1985), Australian National University (1987), TH Darmstadt (1989) oraz G. Mason Univ. (1990-1991). Modelowanie, identyfikacja, estymacja, diagnostyka, adaptacja, teoria sterowania, przetwarzanie sygnałów, sztuczna inteligencja, inżynieria sterowania i informatyka. Ponad 10 książek (WNT, Springer, PWNT), ponad 90 artykułów, 300 referatów.

e-mail: kova@pg.gda.pl

Inż. Jan KLIMCZAK

W 2012 roku uzyskał tytuł inżyniera w Katedrze Systemów Decyzyjnych na Wydziale Elektroniki, Telekomunikacji i Informatyki Politechniki Gdańskiej w Informatyce. Pracuje na stanowisku Kierownika Projektu w firmie Sii Sp. z o.o. na rzecz Nordea Bank. Wyróżniony na apelu 80-lecia KPW Gdynia (2005). W 2012 roku zajął III Miejsce w Ogólnopolskim Konkursie na Najlepszą Symulację Komputerową SimComp 2012 na AGH w Krakowie.

e-mail: klimczakjan@gmail.com

Streszczenie

W niniejszej pracy rozważa się zagadnienie wykorzystania inteligentnego systemu nawigowania oraz przeglądania serwisu Internetowego z wykorzystaniem głosu. Przeanalizowano miejsca i mechanizmy występowania problemów, które są krytyczne dla działania systemu. Zaprezentowano przykładową pełną implementację rozważanego systemu, która pozwala też na wyznaczenie kierunków dalszego jego rozwoju.

Słowa kluczowe: inteligentny system sterowania, system generowania i rozpoznawania mowy, Internet, sterowanie głosem.

An intelligent web-navigation system controlled by voice

Abstract

We consider the problem of using intelligent service navigation and browsing the Internet using a voice-controlled system. The issues critical for system operation are analyzed. After presenting general characteristics, there are described the system architecture, its modular-layered structure, initialization, the way of connecting a customer, and the methods for generating speech and recognizing voice. A short description of an exemplary complete implementation of the intelligent voice-controlled navigation through the website is given. Initial tests have shown the integral and stable operation of the whole system. The applied libraries, used for speech recognition and synthesis, are though not perfect. The quality of the generated speech needs further improvement, and the speech recognition system has a high error rate. Though the pace of development of the speech libraries is not too high, they are constantly being developed, and thus they can be successively updated.

Keywords: decision-making systems, speech recognition and synthesis, Internet, voice control.

1. Wprowadzenie

Wykorzystując moduł mowy uzyskać można bardzo naturalny interfejs, który umożliwia użytkownikowi przyjazne sterowanie systemem w sposób bardzo naturalny.

Możemy wyróżnić dwa główne obszary systemu mowy: rozpoznawanie mowy (ang. *Speech Recognition*) oraz syntezę mowy (ang. *Speech Synthesis*). Proces rozpoznawania mowy polega na przetworzeniu zarejestrowanego sygnału mowy na postać tekstową (ang. *speech-to-text*). Synteza mowy jest procesem odwrotnym, który polega na zamianie tekstu na odpowiednio generowany sygnał mowy (ang. *text-to-speech*, TTS) [1].

Niniejszy projekt, na podstawie przeglądu zagadnień, scala istniejące rozwiązania, które umożliwiają sterowanie portalem internetowym za pomocą głosu w języku angielskim. Użytkownik otrzymuje możliwość wydawania komend głosowych, a także zapytań odnośnie zawartości portalu. W odróżnieniu od wielu systemów istniejących na rynku, dzięki wykorzystaniu wtyczki Adobe Flash Player (obecnie instalowanej na większości komputerów), prezentowany projekt jest systemem typu klient-serwer, który nie wymaga od użytkownika instalowania dodatkowego oprogramowania po swojej stronie. Rozpoznawanie i generowanie mowy oraz wszystkie obliczenia systemu wykonywane są po stronie serwera.

W porównaniu do implementacji [2], wychodzimy o krok na przód dostarczając system transparentny dla użytkownika. System wtopiony jest w dany portal i jego zastosowanie ogranicza się wyłącznie do odpowiednio przygotowanych stron internetowych. Takie rozwiązanie – przy ograniczeniu zasięgu działania systemu do wybranego obszaru – pozwoliło na znaczne rozszerzenie jego możliwości.

Ze względu na złożoność istniejących stron internetowych, w tym rozwiązaniu zdecydowaliśmy się utworzyć własny język znaczników na wzór specyfikacji HTML5 Speech, za pomocą których sam autor decyduje, jakie treści i w jaki sposób powinny być interpretowane. Dzięki temu uzyskuje się m.in. możliwość nawigacji poprzez użycie konkretnych komend głosowych – np. "Go to Laboratories" w miejsce komendy np. "Go to 23", jak ma to miejsce w większości dostępnych systemów tego typu, np. w systemie [2] lub we wbudowanych mechanizmach rozpoznawania mowy w przeglądarkach internetowych. Udało sie w ten sposób usunąć wszelkie 'dodatkowe' oznaczenia stron używane do nawigacji za pomocą komend głosowych. Co więcej, w ten sposób udało się uzyskać możliwość zaindeksowania komend głosowych dla technicznie trudno dostępnych miejsc, np. dla menu rozwijanych, tak jak to uczyniono w implementacji [3].

Na tym różnice się nie kończą. Prezentowany system pozostaje z użytkownikiem w interakcji i na ustaloną komendę głosową odczytuje nam zawartość portalu w odpowiednio zaplanowany sposób. Może np. z gąszczu tekstu, linków i grafiki przeczytać nam opracowanie w taki sposób, w jaki uczynił by to człowiek czytający tytuł artykułu i jego treść z pominięciem wszystkich innych elementów nie związanych z nim, np. reklam śródtekstowych. Możemy także wydać komendę, która przedstawi nam zawartość danej strony w skrócie jeszcze zanim zdecydujemy się na jej odsłuchanie w całości, np. w odpowiedzi system wygeneruje nam wypowiedź w rodzaju "This article is about ...".

Głównym wyróżnikiem niniejszego systemu jest zastosowanie modułu mózgu [4], który interpretując wydawane komendy głosowe oraz śledząc sposób nawigacji po portalu, oddziałuje na sposób interakcji systemu z użytkownikiem (zatem wydanie komendy głosowej nie musi wywoływać tej samej reakcji). Moduł mózgu wyposażony jest w 4 podstawowe stany ("happines", "satisfaction", "excitement", "joy"), które są nieustannie aktualizowane i na podstawie których system kieruje swoim zachowaniem. Np. zmiana intonacji, tempa i modulacji generowanego głosu, udzielenie odpowiedzi adekwatnej do danego stanu (np. na zapytanie "How are You ?"), może wywołać zupełnie różne odpowiedzi.

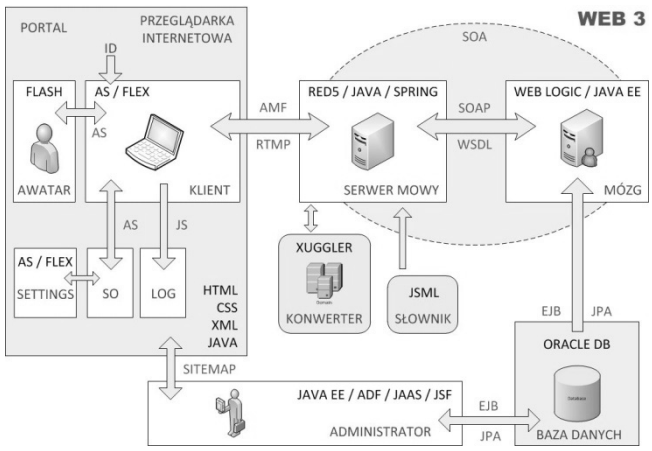
Moduł ten odpowiada także za przełączanie stron (komenda "Please go to next page"): W zależności od preferencji użytkownika na podstawie historii jego nawigacji kieruje użytkownika na najbardziej odpowiednią stronę i zmienia wizualną postać Inteligentnego Awatara (ang. *IntelligentAvatar*) [5].

Awatar stanowi graficzną emanację stanu modułu mózgu w postaci ludzkiej. Może się zmieniać od postaci wesołej, zadowolonej do smutnej i gniewnej. Awatar wita nas przy pierwszym wejściu na Portal Internetowy, a później udziela nam odpowiedzi na zadawane przez nas pytania, które są interpretowane przez moduł mózgu systemu. Awatar przystosowany jest do rozszerzenia do postaci *Dictobot*'a [6], autonomicznego agenta, opierającego się na matematycznym modelu psychologii osobowości człowieka [4], który w zależności od stanu emocjonalnego, w jakim się znajduje, w odpowiedni sposób interpretuje usłyszane kwestie oraz formułuje własną wypowiedź, adekwatną do stanu 'rozmowy' z użytkownikiem (np. zwracać na siebie uwagę, domagać się o kontakt). A to oznacza, że interakcja lub odpowiedź na pytanie, nie polega na "sztywnej" bazie wiedzy.

Takie cechy szczególnie odróżniają prezentowane podejście od innych dostępnych systemów. Nie jest to jedynie 'pilot' do sterowania nawigacją po Portalu Internetowym. Jest to raczej humanoidalny system, z którym przyjemniej się współpracuje i który po rozwinięciu do postaci Dictobot'a mógłby pomagać użytkownikowi w nawigacji, a nawet troszczyć się o niego (np. pytając "Do you mind this or those ?" lub udzielać stosownych porad "It's time to go to bed"). Choć brzmi to teraz trochę jak *science-fiction*, takie rozwiązania mogą się w przyszłości upowszechnić, ułatwiając życie nam oraz osobom upośledzonym.

2. Architektura systemu

Architekturę systemu przedstawia rys. 1. System jest silnie rozproszony i dzieli się na niezależnie zarządzane moduły.



Rys. 1. Ogólna architektura systemu
Fig. 1. General architecture of the system

Wydzielono zarządzaną przez użytkownika część kliencką, która jest środowiskiem pracy złożonym z modułów:

- *Klient* (*KLIENT*) – odpowiada za dwustronną komunikację z serwerem, przydzielanie uprawnień do mikrofonu i głośników, wygenerowanie unikalnego identyfikatora dla klienta oraz nagrywanie i odtwarzanie głosu.
- *Awatar* (*AWATAR*) – wizualizuje stan modułu mózgu dla aktualnego użytkownika na podstawie jego lokalnego stanu (o wykresie słupkowym). Animacja Awatara (rys. 2) dokonuje się w sposób ciągły. Awatar wyposażony jest w samoistną świadomość z mechanizmem "zapominania" i normalizacji stanu, który działa wyłącznie po stronie klienta doprowadzając emocje do stanu neutralnego.

Rys. 2. Animacja postaci Awatara w zależności od stanu (emocji)
Fig. 2. Animation of the Avatar based on its emotions

- *Moduł pamięci klienta* (*SO*) – Przechowuje aktualny stan Awatara, unikalny dla każdego użytkownika. Jest on przypisany na stałe do danego komputera i przeglądarki. Przechowywany jest na komputerze klienta. Dotyczy nie tylko danej sesji użytkownika (*de facto* nie wymaga logowania się do systemu), ale także poprzednich odwiedzin portalu internetowego.
- *Moduł logów* (*LOG*) – konsola logowania systemu, w której użytkownik monitoruje aktualny jego stan.

Część serwerowa podzielona jest na dwie platformy:

- *Serwer mowy* – serwer czasu rzeczywistego, kluczowy element całego systemu. Bezpośrednio z nim komunikuje się klient-użytkownik systemu. Serwer ten determinuje interakcję z użytkownikiem. Tu mieszczą się konfiguracje systemów rozpoznawania i syntezy mowy w formatach zdefiniowanych przez specyfikacje bibliotek FreeTTS [7] oraz Sphinx4 [8].
- *Serwer aplikacji* - serwer OracleWebLogic 11g, na którym działa moduł mózgu systemu, część administracyjna oraz portal demonstracyjny.

Serwer mowy posiada architekturę wielowątkową (ang. *Multithreading*). Operacje realizowane są w oparciu o zdarzenia (ang. *Events*). Dotyczy to obsługi komunikacji serwera z klientem podczas wymieniania komunikatów, konwersji plików audio oraz obsługi wymiany danych audio w czasie rzeczywistym.

Klient łączy się z częścią serwerową za pośrednictwem protokołów RTMP [9] i AMF (ang. *Action Message Format*). Za pomocą protokołu RTMP przesyłany w czasie rzeczywistym jest wyłącznie sygnał audio. Natomiast protokół AMF służy jedynie do wymiany komunikatów pomiędzy serwerem a klientem. Komunikaty te zawierają: stan modułu mózgu Awatara, komendy tekstowe przesyłane do serwera, informacje zwrotne dla klienta, na którego czeka wiadomość do odsłuchania, komunikaty logów ze strony serwera, itp. Serwer mowy komunikuje się z modułem mózgu za pośrednictwem protokołu SOAP (ang. *Simple Object Access Protocol*) w technologii rozproszonej SOA (ang. *Service-oriented architecture*).

W części serwerowej wyróżniamy następujące moduły:

- *Moduł mózgu* – odpowiada za logikę aplikacji. Komunikuje się z systemem mowy, od którego dostaje komendę użytkownika, przetwarza ją, po czym zwraca akcję do wykonania przez klienta w portalu internetowym. Po uzyskaniu nowych danych oraz przetworzeniu posiadanych danych, moduł ten decyduje o interakcji w portalu internetowym za pomocą serwera mowy. Na podstawie aktualnego stanu, moduł mózgu generuje odpowiedź, która wywołuje (odmienne) zachowanie się Awatara oraz zmienia modulację jego głosu.
- *Modułu klienta mózgu* – komunikuje serwer mowy z modułem mózgu poprzez protokół SOAP przy wykorzystaniu dokumentu WSDL (ang. *Web Services Description Language*) w technologii rozproszonej SOA.
- *Moduł rozpoznawania głosu* – rozróżnia komendy głosowe na podstawie listy słów i komend rozpoznawalnych przez system w formacie JSGF (ang. *Java Speech Grammar Format*) [10] po czym konwertuje je do postaci tekstu.
- *Moduł syntezy mowy* – generuje głos na podstawie dostarczonego tekstu.

- *Moduł konwersji audio* – konwertuje nadsyłane dane audio. Wykorzystuje bibliotekę Xuggler do konwersji formatów audio. Standardowo serwer Red5 nagrywa dane audio w formacie .flv (ang. *Flash Video*), które są następnie konwertowane do formatu .wav (ang. *Wave form audio format*) dla modułu syntezy mowy po czym konwertuje się go do formatu .mp3 (ang. *MPEG-1/MPEG-2 Audio Layer 3*), który pozwala na odtwarzanie w czasie rzeczywistym przez klienta.
- *Moduł administratora* – służy do zapełniania bazy danych. Odczytuje zawartość portalu na podstawie jego mapy strony (ang. *Sitemap*) oraz wcześniej przygotowanych znaczników mowy (ang. *Tags*). Moduł ten jest całkowicie niezależny od pozostałej części systemu, zapewnia możliwość podglądu wygenerowanych danych, wygenerowania nowych danych oraz edycji komunikatów przesyłanych użytkownikowi.

3. Algorytm działania systemu

System posiada budowę modułową i warstwową, co pozwala wyizolować poszczególne jego algorytmy.

3.1. Przygotowanie systemu mowy

Podczas startu systemu inicjalizowany jest system mowy i sam serwer czasu rzeczywistego. Inicjalizacja serwera mowy polega na załadowaniu i wybraniu głosu, rezerwacji zasobów sprzętowych niezbędnych przy generowaniu mowy i przypisaniu generatora do systemu mowy. Później inicjalizowany jest system rozpoznawania mowy. W pierwszej kolejności następuje skonfigurowanie systemu, który wczytuje listę słów w formacie JSGF i komend do rozpoznania przy uwzględnieniu danej rozdzielczości sygnału audio, model akustyczny oraz inne ustawienia wykorzystywane podczas rozpoznawania mowy.

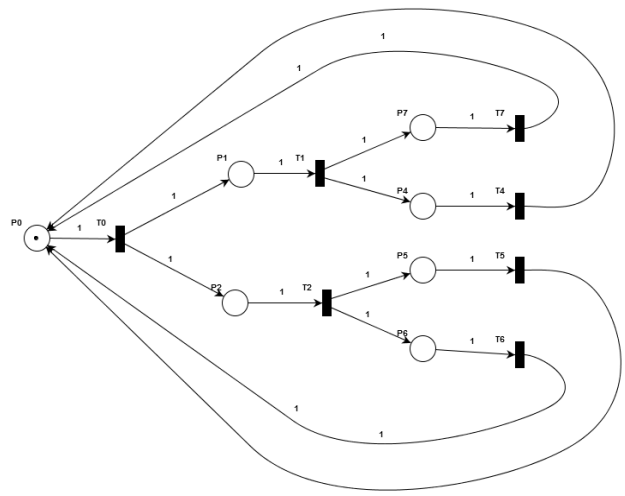
3.2. Podłączenie klienta

Interakcja po stronie użytkownika następuje już w momencie otworzenia strony portalu internetowego. Po stronie portalu zostaje wygenerowany unikalny identyfikator użytkownika, który wykorzystywany jest w połączeniach klienta z serwerem.

Następnie zostaje załadowany klient w technologii Flex poprzez wtyczkę przeglądarki internetowej Adobe Flash. Klient w momencie startu wprowadza postać graficzną Awatara. Potem następuje inicjalizacja klienta, która uwzględnia utworzenie specjalnej wersji systemu logowania aplikacji klienta, która wysyła logi aplikacji dla użytkownika za pośrednictwem języka Java Script. Kolejno utworzony jest lub wczytany współdzielony obiekt (ang. *Shared Object*) ustawień danego klienta. W następnej kolejności zostaje zarezerwowany mikrofon do wykorzystania w systemie. W tym momencie następuje wczytanie pozostałych ustawień klienta zapisanych w obiekcie współdzielonym. Po wczytaniu ustawień odtwarzany jest stan Awatara z obiektu współdzielonego. W następnym kroku przygotowuje się klienta do wymiany komunikatów z serwerem. Po tych czynnościach Awatar zostaje ożywiony i poddany normalizacji (powrotowi do stanu normalnego).

Po przygotowaniu klienta następuje połączenie klienta z serwerem. Po wysłaniu żądania połączenia z serwerem, serwer podejmuje próbę połączenia z klientem w osobnym wątku. Klient wysyła swój identyfikator do serwera. W pierwszym momencie po stronie serwera następuje weryfikacja, czy łączący się klient czasem nie jest już połączony z serwerem w innym żądaniu (np. w innej zakładce przeglądarki lub w wyniku przejścia na inną stronę portalu internetowego). Jeżeli klient jest już połączony to następuje scalenie jego sesji z obecnie zapisaną w liście klientów aktualnie połączonych, w przeciwnym razie klient zostaje dodany do listy nowych klientów aktualnie połączonych z systemem. W przypadku sukcesu w osobnym wątku zostaje wygenerowana wiadomość powitalna dla klienta.

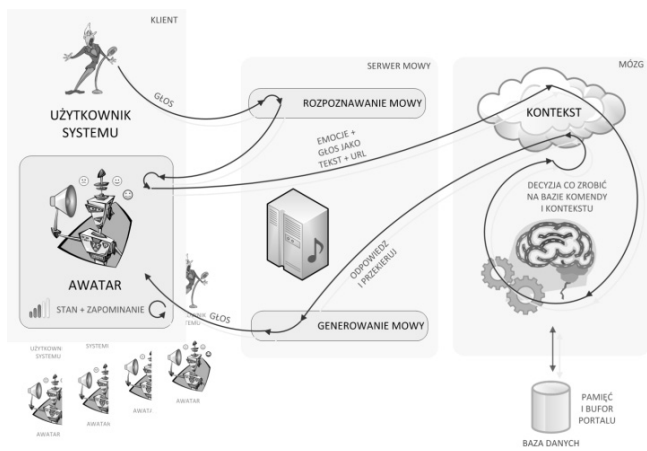
Rysunek 3 przedstawia sposób wykorzystania wątków podczas inicjalizacji klienta. P0 oznacza klienta, T0 reprezentuje podłączenie klienta do serwera. Każdy klient zostaje obsłużony w osobnym wątku. T1 oraz T2 obrazują wygenerowanie wiadomości powitalnej, która podawana jest także do osobnych wątków. Wszystkie te komunikaty wracają z powrotem do klientów.



Rys. 3. Wykorzystanie wątków podczas inicjalizacji klienta
Fig. 3. Use of threads during the client initialization stage

3.3. Generowanie mowy i jej odsłuchiwanie

W zależności od sposobu implementacji modułu mózgu, przy wykorzystaniu obiektu złożonego można uzyskać rozmaity wynik wygenerowanej mowy (w zależności od aktualnego stanu Awatara). W przypadku prostego tekstu zawsze otrzymamy wygenerowaną mowę do odsłuchania. Schemat generowania komendy głosowej został przedstawiony na rys. 4.



Rys. 4. Wygenerowanie odpowiedzi na podstawie komendy głosowej
Fig. 4. Generating an answer based on a voice command

Na początku tworzy się obiekt zapytania, który zawiera komendę tekstową oraz stan Awatara. Poprzez klienta obiekt zapytania jest przekazywany do modułu mózgu za pośrednictwem protokołu SOAP. W omawianym systemie następuje analiza nadesłanej komendy głosowej po czym następuje przygotowanie odpowiedzi zwrotnej, która zawiera tekst odpowiedzi pobrany z bazy danych oraz wartości zmiany stanu.

Równolegle, na podstawie wcześniej zapisanego identyfikatora klienta, który zgłosił żądanie na wygenerowanie mowy, rozpoczyna się proces udźwiękowienia, tzn. wygenerowania pliku zawiera-

jącego mowę. Po wygenerowaniu pliku mowy i przekonwertowaniu go do formatu umożliwiającego jego odtworzenie w czasie rzeczywistym wysłana zostaje komenda *odtwórz wiadomość* wraz z nazwą pliku do odtworzenia dla klienta zgłaszającego prośbę o wygenerowanie mowy. Klient otrzymawszy taką wiadomość odtwarza ją w czasie rzeczywistym.

3.4. Rozpoznanie komendy głosowej

Rozpoznanie komendy głosowej rozpoczyna się w momencie wciśnięcia i przytrzymania przycisku "Speak". Przy wciśniętym przycisku następuje nagranie materiału audio/video na serwerze w czasie rzeczywistym pod wysłaną nazwą klienta. Podczas tej operacji serwer przechwytuje materiał i zapisuje go na dysku komputera. W momencie puszczenia przycisku przez użytkownika wysyłany jest komunikat o ukończeniu nagrywania do serwera. W nowym wątku po stronie serwera następuje konwersja pliku a następnie rozpoznanie komendy i wykonanie akcji opisanej w poprzednim podrozdziale.

4. Implementacja systemu

Utworzony system został w głównej mierze oparty na standardach Open Source. Wszystkie jego moduły zostały od podstaw utworzone na potrzeby prezentowanej platformy. Do ich wykonania zostały wykorzystane istniejące już biblioteki: rozpoznawania mowy: *Sphinx4*, syntezy mowy *FreeTTS*, konwersji audio *Xuggler*, framework części serwerowej *Spring Source Framework* oraz biblioteki *Red5*, parser znaczników HTML *JSoup* oraz systemu logowania *Apache Logger*.

Rys. 5. Widok portalu demonstracyjnego
Fig. 5. Overview of the demo-portal

Cały system został wytworzony w technologii JavaEE / Adobe Flex (Action Script). Technologia Adobe Flex została wykorzystana do utworzenia klienta, natomiast część serwerowa została wykonana w technologii Java. Fizycznie część ta podzielona jest na dwa serwery: serwer mowy jest serwerem czasu rzeczywistego (Red5 RC 1.0), który opiera się na platformie Apache Tomcat 6, który obsługuje port RTMP. To bezpośrednio z nim komunikuje się klient (a w zasadzie użytkownik systemu). Serwer ten wydziela dalszą interakcję z użytkownikiem. Na serwerze tym utworzone zostały moduły generowania i syntezy mowy, klienta modułu mózgu, obsługi i konwersji danych audio przesyłanych w czasie rzeczywistym. Także w tym miejscu znajduje się lista słów i komend rozpoznawalnych przez system w formacie JSGF (ang. Java Speech Grammar Format) oraz konfiguracja systemów rozpoznawania i syntezy mowy w formatach zdefiniowanych przez specyfikacje bibliotek FreeTTS i Sphinx4. Drugim serwerem jest Oracle WebLogic 11g, na którym działa moduł mózgu systemu, część administracyjna oraz portal demonstracyjny.

Budowa platformy jest modułowa, umożliwiająca niezależny rozwój i wymianę każdego z modułów

Widok wykonanego portalu demonstracyjnego działanie systemu został przedstawiony na rys. 5.

5. Wnioski

Pierwsze testy integralne całego systemu wykazały jego stabilną pracę. Jednak wykorzystane biblioteki rozpoznawania i syntezy mowy nie są doskonałe. Jakość wygenerowanej mowy pozostawia wiele do życzenia, a sam system rozpoznawania mowy charakteryzuje się wysoką stopą błędów, zaś wydawane komendy są często źle interpretowane. Choć tempo rozwoju bibliotek generowania i syntezy mowy nie jest zbyt wysokie, są one jednak stale rozwijane. Zatem, w miarę pojawiania się lepszych wersji, mogą być one sukcesywnie aktualizowane w opisanym systemie inteligentnej nawigacji sterowanej głosem.

6. Literatura

[1] O'Shaughnessy D.: Interacting With Computers by Voice: Automatic Speech Recognition and Synthesis, vol. 91, no. 9. PDF: (http://www.hoviat.dpsh.net/download/maghale-signal/15.pdf), IEEE, Montreal (Canada) 2003.
[2] Mięsikowska M.: Speech command based application enabling Internet navigation, PDF: (http://www.knws.uz.zgora.pl/history/pdf/knws_07_miesikowska_m_87-89.pdf), PAK vol. 53, str. 87-89, nr 5/2007.
[3] Brøndsted T., Aaskoven E.: Voice-controlled Internet Browsing for Motor-handicapped Users. http://kom.aau.dk/~tb/articles/INDTAL Interspeech2005.php, From Proc.: Interspeech - Eurospeech - 9th European Conference on Speech Communication and Technology, Lisboa 2005.
[4] Kowalczuk Z., CzubenkoM.: Intelligent decision-making system for autonomous robots. Int. Journal of Applied Mathematics and Computer Science, vol. 21, no. 4, pp. 671-684, 2011.
[5] Huang Z., Eliëns H., Vissser C.: Programmability of Intelligent Agent Avatars, PDF: (http://wasp.cs.vu.nl/wasp/papers/avatar.pdf), Vrije University of Amsterdam, Montreal (Canada) 2001.
[6] Kowalczuk Z., Czubenko M.: Dictobot –An autonomous agent with the ability to communicate. Technologie Informacyjne, str. 87-92. ZeszytyNaukowe WETI, Politechnika Gdańska, 2011.
[7] FreeTTS 1.2 - A speech synthesizer written entirely in the Java programming language. http://freetts.sourceforge.net/docs/index.php, 2011.
[8] Sphinx-4 - A speech recognizer written entirely in the Java programming language. http://cmusphinx.sourceforge.net/sphinx4/, 2011.
[9] Real-Time Messaging Protocol (RTMP) specification. http://www.adobe.com/devnet/rtmp.html, 2011.
[10] Java Speech API. http://java.sun.com/products/java-media/speech/, 2011.

otrzymano / received: 06.12.2012
przyjęto do druku / accepted: 03.06.2013 *artykuł recenzowany / revised paper*