IMPLEMENTASI SISTEM PENGENALAN SUARA MENGGUNAKAN SAPI 5.1 DAN DELPHI 5

Agustinus Noertjahyana, Rudy Adipranata
Fakultas Teknologi Industri, Jurusan Teknik Informatika – Universitas Kristen Petra
email : agust@peter.petra.ac.id, rudya@yahoo.com

ABSTRAK: Sekarang ini pengenalan pembicaraan pada aplikasi komputer bukan merupakan hal yang baru. Banyak penelitian yang telah dan sedang dilakukan untuk mendapatkan pengenalan pembicaraan yang cepat dan akurat. Salah satu yang terkenal adalah penelitian yang dilakukan oleh Microsoft Corporation yang dikembangkan untuk sistem operasi Windows. Selain mengembangkan engine pengenalan pembicaraan, Microsoft juga mengembangkan standard untuk pengenalan pembicaraan yaitu SAPI (Speech Application Programming Interface) yang sekarang sudah mencapai versi 5.1.

Dengan adanya SAPI memungkinkan pembuat aplikasi untuk mengimplementasikan sistem pengenalan pembicaraan dengan menggunakan *engine* sesuai yang diinginkan dan dapat mengganti penggunaan dari satu *engine* ke *engine* yang lain tanpa merubah aplikasi yang telah dibuat.

Penelitian ini mengimplementasikan sistem pengenalan pembicaraan dengan menggunakan SAPI 5.1, Microsoft Speech Engine dan bahasa pemrograman Delphi 5 yang digunakan untuk melakukan diktasi berbahasa Inggris pada aplikasi berbasis teks.

Kata kunci: Pengenalan suara, SAPI.

***ABSTRACT**: Nowadays, speech recognition in computer application is not new topic. Speech recognition research is still in active area with purpose to get fast and accurate result. One of the famous speech recognition research is done by Microsoft Corporation which developed for Windows platform. Also Microsoft develop standard for speech recognition engine, namely SAPI (Speech Application Programming Interface) which now get into version 5.1.*

Using SAPI, programmer can implement speech recognition using any kind of speech recognition engine and also can change the engine without consider for changing the program code.

In this paper, we implement speech recognition system using SAPI 5.1, Microsoft Speech Engine and Delphi 5 as programming language to create text application for English dictation.

Keywords: *Speech Recognation, SAPI.*

1. PENDAHULUAN

Sistem pengenalan pembicaraan pada aplikasi komputer sekarang sudah bukan merupakan hal yang baru. Banyak penelitian yang dilakukan untuk terus meningkatkan kemampuan pengenalan pembicaraan. Diantaranya yang cukup terkenal adalah penelitian yang dilakukan oleh Microsoft Corporation. Microsoft telah mengembangkan sistem pengenalan pembicaraan yang diimplementasikan pada sistem operasi Windows. Pada sistem ini juga dikembangkan standard interface SAPI (Speech Application Programming Interface) yang saat ini telah mencapai versi 5.1. Dengan adanya SAPI memungkinkan pembuat aplikasi untuk mengimplementasikan sistem pengenalan pembicaraan dengan menggunakan *engine* yang berbeda tanpa merubah aplikasi yang telah dibuat.

Penelitian ini mengimplementasikan sistem pengenalan pembicaraan dengan menggunakan SAPI 5.1, *engine* Microsoft Speech Engine dan bahasa pemrograman Delphi 5 yang digunakan untuk melakukan diktasi berbahasa Inggris pada aplikasi berbasis teks.

2. SISTEM PENGENALAN PEMBICARAAN

Terdapat dua macam mode pada sistem pengenalan pembicaraan yaitu:

1. Mode diktasi. Pada mode ini pengguna komputer dapat mengucapkan kata/ kalimat yang selanjutnya akan dikenali oleh komputer dan diubah menjadi data teks. Kemungkinan jumlah kata yang dapat dikenali dibatasi oleh jumlah kata yang telah terdapat pada database. Pengenalan mode diktasi merupakan *speaker dependent*. Keakuratan pengenalan mode ini bergantung pada pola suara dan aksen pembicara serta pelatihan yang telah dilakukan.
2. Mode *command and control.* Pada mode ini pengguna komputer mengucapkan kata/kalimat yang sudah terdefinisi terlebih dahulu pada database dan selanjutnya akan digunakan untuk menjalankan perintah tertentu pada aplikasi komputer. Jumlah perintah yang dapat dikenali tergantung dari aplikasi yang telah mendefinisikan terlebih dahulu pada database jenis-jenis perintah yang dapat dieksekusikan. Mode ini merupakan *speaker independent* karena jumlah kata yang dikenali biasanya terbatas sekali dan ada kemungkinan pembicara tidak perlu melakukan pelatihan pada sistem sebelumnya.

Terdapat empat proses utama pada sistem pengenalan pembicaraan, baik pada mode diktasi ataupun *command and control* yaitu : pemisahan kata, ketergantungan terhadap pengguna, pencocokan kata dan perbendaharaan kata.

Pemisahan kata ialah proses untuk memisahkan suara yang diucapkan oleh pengguna menjadi beberapa bagian. Masing-masing bagian bisa berupa kalimat ataupun hanya sebuah kata. Terdapat tiga macam metode yang dapat digunakan pada proses pemisahan kata ini yaitu : *discrete speech*, *word spotting* dan *continuous speech*. Pada *discrete speech*, pengguna diharuskan mengucapkan kalimat secara terpenggal dengan adanya jeda sejenak diantara kata. Jeda tersebut digunakan oleh sistem untuk mendeteksi awal dan akhir sebuah kata. *Discrete speech* ini mempunyai kelebihan yaitu sedikit *resource* (memori komputer, waktu proses) yang digunakan oleh sistem untuk mendeteksi suara, tetapi mempunyai kelemahan yaitu ketidaknyamanan pengguna dalam mengucapkan sebuah kalimat. Pada *word spotting*, dalam sebuah kalimat yang diucapkan pengguna, sistem hanya mendeteksi kata yang terdapat di dalam perbendaharaan yang dimilikinya, dan mengabaikan kata-kata lain yang tidak dimilikinya. Sehingga walau pengguna mengucapkan kalimat yang berbeda tetapi di dalam kalimat tersebut terdapat sebuah kata yang sama dan terdapat di perbendaharaan sistem, maka hasil pengenalan akan sama. Kelemahan metode ini ialah besar kemungkinan sistem akan melakukan kesalahan arti pengenalan dalam bentuk kalimat. Tetapi metode ini mempunyai kelebihan yaitu pengguna dapat mengucapkan kalimat secara normal tanpa harus berhenti diantara kata. Pada metode *continuous speech*, sistem akan mengenali dan memproses setiap kata yang diucapkan. Metode ini akan menghasilkan keakuratan dalam mengenali ucapan pengguna. Tetapi di samping itu metode ini memerlukan *resource* yang besar dalam prosesnya. Pada metode ini, sistem harus dapat mendeteksi awal dan akhir setiap kata dalam kalimat tanpa adanya jeda diantara kata-kata tersebut, dan setelah berhasil memisahkan kata, langkah selanjutnya adalah mencocokkan dengan perbendaharaan kata yang dipunyainya.

Sistem pengenalan pembicaraan mempunyai beberapa sifat dilihat dari ketergantungan terhadap pengguna yaitu *speaker dependent*, *speaker independent* dan *speaker adaptive*. Pada *speaker dependent*, sistem membutuhkan pelatihan untuk setiap pengguna yang akan menggunakan sistem tersebut. Sistem tidak akan bisa mengenali pengguna yang belum pernah melakukan pelatihan. Pada *speaker independent*, pengguna tidak perlu melakukan pelatihan sebelum dapat menggunakan sistem, karena sistem mampu mengenali suara semua pengguna tidak tergantung warna suara dan dialek yang digunakan. *Speaker adaptive* merupakan perpaduan dari *speaker dependent* dan *speaker independent*, dimana pengguna tidak perlu melakukan pelatihan dan keakuratan pengenalan sistem akan makin meningkat jika pengguna yang sama bekerja terus menerus selama beberapa waktu tertentu.

Pencocokan kata adalah proses untuk mencocokkan kata ucapan yang berhasil diidentifikasi dengan basis data yang dipunyai oleh sistem. Terdapat dua metode yang dapat dipakai pada proses pencocokan kata ini, yaitu : *whole-word matching* dan *phoneme matching*. Pada *whole-word matching*, sistem akan mencari di basis data kata yang sama persis dengan kata hasil ucapan pengguna. Sedangkan pada *phoneme matching*, sistem mempunyai kamus fonem. Fonem ialah bagian terkecil dan unik dari suara yang membentuk sebuah kata.

Perbendaharaan kata ialah bagian terakhir dalam sebuah sistem pengenalan pembicaraan. Terdapat dua hal yang perlu diperhatikan pada perbendaharaan kata, yaitu ukuran dan keakuratan. Jika perbendaharaan kata berjumlah banyak maka sebuah sistem akan mudah dalam melakukan pencocokan kata, tetapi dengan makin meningkatnya jumlah perbendaharaan kata, maka jumlah kata yang mempunyai ucapan hampir sama juga meningkat, dimana hal ini menurunkan keakuratan pengenalan. Dan sebaliknya, jika sebuah sistem mempunyai perbendaharaan kata sedikit, maka keakuratan pengenalan akan tinggi karena sedikitnya kata yang hampir sama, tetapi akan semakin banyak kata yang tidak terkenali. Untuk sistem pengenalan pembicaraan dengan mode *command and control*, akan lebih baik jika menggunakan jumlah perbendaharaan kata sedikit (kurang dari 100 kata), tetapi untuk mode diktasi akan membutuhkan jumlah perbendaharaan kata yang banyak.

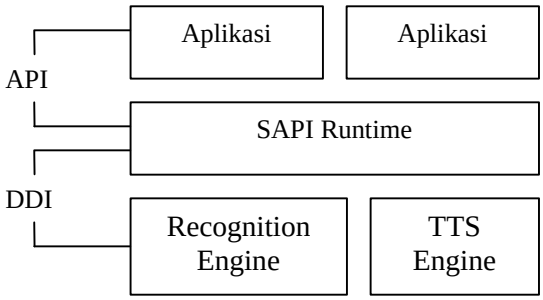
Pada penelitian ini aplikasi yang dikembangkan adalah aplikasi berbasis teks untuk menerima diktasi berbahasa Inggris. Sehubungan dengan aplikasi tersebut, maka sifat sistem pengenalan pembicaraan yang digunakan adalah *continuous speech, dependent speaker*.

3. ARSITEKTUR SPEECH APPLICATION PROGRAMMING INTERFACE (SAPI 5.1)

SAPI 5.1 terdiri dari 2 antar muka yaitu *application programming interface* (API) dan *device driver interface* (DDI).

3.1 *Application Programming Interface (API)*

Pada sistem pengenalan pembicaraan, aplikasi akan menerima even pada saat suara yang diterima telah dikenali oleh *engine*.



Gambar 1. Blok Diagram Arsitektur SAPI

Komponen SAPI yang akan menghasilkan even ini diimplementasikan oleh antar muka ISpNotifySource. Lebih spesifik, SAPI menggunakan SetNotifySink, yaitu aplikasi akan meneruskan pointer IspNotifySink ke ISpNotifySource:: SetNotifySink. ISpNotifySource::SetNotifySink ini akan menerima pemanggilan melalui IspNotifySink:: Notify ketika terdapat satu atau lebih even yang menyatakan bahwa aplikasi dapat mengambil data. Biasanya aplikasi tidak mengimplementasikan ISpNotifySink secara langsung tetapi menggunakan CoCreate Instance untuk membuat obyek IspNotify Sink, yang diimplementasikan oleh komponen CLSID_SpNotify. Obyek ini menyediakan antar muka ISpNotifyControl.

Tetapi antar muka ISpNotifySource dan ISpNotifySink hanya menyediakan mekanisme untuk notifikasi dan tidak ada even yang ditimbulkan oleh notifikasi tersebut. Dengan menggunakan obyek IspEvent Source, aplikasi dapat menerima informasi tentang even yang ditimbulkan oleh notifikasi. ISpEventSource juga menyediakan mekanisme untuk menyaring dan membuat antrian even. Biasanya aplikasi tidak menerima notifikasi dari IspEvent Source sampai terjadi pemanggilan terhadap ISpEventSource::SetInterest untuk menentukan even mana yang akan menghasilkan notifikasi dan even mana yang berulang sehingga harus dimasukkan ke daftar antrian. Even diidentifikasi dengan menggunakan tanda SPEVENTENUM.

Ketika aplikasi menerima notifikasi, ada kemungkinan terdapat informasi yang sama pada beberapa even. Dengan memanggil ISpEventSource::GetInfo, maka variabel anggota ulCount akan mengembalikan nilai yang berupa struktur SPEVENT SOURCEINFO yang didalamnya terdapat jumlah even yang mempunyai informasi yang sama. Dengan menggunakan IspEvent Source::GetEvents, aplikasi akan mengeluarkan sejumlah struktur SPEVENT, di mana masing-masing mempunyai informasi tentang even tertentu.

Langkah pertama pembuatan sistem pengenalan pembicaraan adalah pembuatan antar muka utama bagi aplikasi. Antar muka utama yang digunakan oleh aplikasi ialah ISpRecoContext. Antar muka ini merupakan sebuah ISpEventSource, yang berarti ISpRecoContext tersebut adalah antar muka aplikasi untuk menerima notifikasi dari sebuah even pengenalan pembicaraan. Aplikasi mempunyai dua pilihan untuk menggunakan *engine* pengenalan pembicaraan (ISpRecognizer). Pilihan pertama ialah menggunakan *shared recognizer* yang memungkinkan untuk berbagi resource dengan aplikasi pengenalan pembicaraan yang lain. Untuk membuat ISpRecoContext untuk *shared* ISpRecognizer, aplikasi hanya memerlukan pemanggilan terhadap CoCreateInstance (COM) pada komponen CLSID_SpSharedRecoContext. Pada kasus ini, SAPI akan mengatur *audio input stream* sesuai dengan pengaturan standard SAPI. *Shared* ISpRecognizer ini paling banyak digunakan pada aplikasi pengenalan pembicaraan secara umum. Sedangkan untuk server aplikasi pengenalan pembicaraan yang besar dan berjalan sendiri pada sebuah sistem serta membutuhkan kemampuan kerja yang tinggi, maka lebih baik menggunakan InProc *engine*. Untuk membuat ISpRecoContext pada InProc ISpRecognizer, maka pertama kali aplikasi harus memanggil CoCreateInstance pada komponen CLSID_SpInprocRecoInstance. Kemudian aplikasi juga harus memanggil ISpRecognizer::SetInput untuk mengatur input suara. Dan terakhir aplikasi harus memanggil ISpRecognizer:: CreateReco Context untuk mendapatkan IspReco Context.

Langkah berikut pembuatan sistem pengenalan pembicaraan adalah pengaturan notifikasi untuk even yang dibutuhkan oleh aplikasi. Karena ISpRecognizer adalah ISpEventSource yang juga merupakan ISpNotifySource, maka aplikasi dapat memanggil salah satu metode IspNotify Source dari ISpRecoContext untuk memberitahukan dimana terdapat sebuah even dari ISpRecoContext.. Kemudian aplikasi juga harus memanggil ISpEventSource:: SetInterest untuk menyatakan even mana yang perlu untuk dinotifikasikan. Even yang paling penting adalah SPEI_RECOGNITION, yang berfungsi untuk menyatakan bahwa ISpRecognizer telah mengenali suara tertentu dari ISpRecoContext.

Langkah terakhir pembuatan sistem pengenalan pembicaraan ini, aplikasi harus membuat, meletakkan di memori (*load*) dan mengaktifkan ISpRecoGrammar, sesuai dengan mode pengenalan pembicaraan, misal : diktasi atau *command and control.* Pertama kali, aplikasi harus membuat ISpRecoGrammar menggunakan IspReco Context::CreateGrammar. Kemudian aplikasi harus meletakkan *grammar* yang telah dibuat tersebut pada memori (*load*) yaitu dengan memanggil ISpRecoGrammar::Load Dictation untuk mode diktasi atau Isp RecoGrammar::LoadCmdxxx untuk mode *command and control*. Untuk mengaktifkan *grammar* sehingga sistem pengenalan pembicaraan dapat bekerja, aplikasi harus memanggil ISpRecoGrammar::Set Dictation State untuk mode diktasi atau IspReco Grammar::SetRuleState/ISpRecoGrammar:: SetRuleIdState untuk mode *command and control*.

Ketika terjadi notifikasi pada saat pengenalan pembicaraan bekerja, maka lParam yang merupakan variabel anggota dari struktur SPEVENT akan menjadi ISpRecoResult yang kemudian digunakan oleh aplikasi untuk dapat menentukan apa yang telah terkenali dan sekaligus menentukan ISpRecoGrammar mana yang harus digunakan.

ISpRecognizer, baik *shared* ataupun InProc, dapat mempunyai ISpRecoContext lebih dari satu dan masing-masing ISpRecoContext dapat menerima notifikasi

sesuai dengan even yang telah didefinisikan. Sebuah ISpRecoContext dapat mempunyai lebih dari satu ISpRecoGrammars di mana masing-masing ISpRecoGrammar tersebut digunakan untuk mengenali tipe *grammar* yang berbeda.

3.2 *Device Driver Interface* (*DDI*)

DDI menyediakan fungsi untuk menerima data suara dari SAPI dan mengembalikan pengenalan frasa pada level SAPI paling dasar. Terdapat dua antar muka yang digunakan oleh DDI yaitu ISpSREngine, yang diimplementasikan oleh *engine* dan ISpSREngineSite yang diimplementasikan oleh SAPI. Antar muka SAPI yaitu ISpSREngineSite juga menyediakan metode untuk memberikan informasi lebih detail mengenai apa yang dikenali oleh *engine*. *Grammars* dan *speakers* menyediakan informasi ke *engine* yang dapat membantu *engine* untuk melakukan pengenalan pembicaraan lebih baik, disamping juga merupakan bagian penting komunikasi yang menghubungkan SAPI dan *speech engine*. Terdapat 2 aspek terakhir yang berhubungan dengan komunikasi antara *engine* dan SAPI yaitu urutan pemanggilan yang terjadi serta masalah *threading.* Salah satu keuntungan menggunakan SAPI 5.1 ini adalah penyederhanaan masalah *threading*.

Engine menyediakan layanan ke SAPI melalui antar muka ISpSREngine. Semua fungsi pengenalan terjadi melalui Isp SREngine::RecognizeStream. Ketika SAPI memanggil ISpSREngine::SetSite, maka SAPI memberikan pointer ke antar muka ISpSREngineSite dimana kemudian *engine* dapat berkomunikasi dengan SAPI selama ISpSREngine::RecognizeStream dieksekusi. SAPI membuat sebuah *thread* ke obyek ISpSREngine dan *engine* tidak boleh meninggalkan ISpSREngine::Recognize Stream sampai terjadi kesalahan atau SAPI sudah terindikasi dengan menggunakan ISpSREngineSite::Read, dimana tidak ada lagi data yang dapat diproses dan *engine* telah selesai melakukan tugasnya.

SAPI memisahkan pembuat *engine* dari kerumitan untuk mengatur peralatan suara secara detail. SAPI menjaga *logical stream* dari *raw audio data* dengan membuat indeks posisi *stream*. Dengan menggunakan indeks posisi *stream*, *engine* dapat melakukan pemanggilan terhadap ISpSREngineSite:: Read untuk menerima *buffer* dari *raw audio data* selama ISpSREngine::Recognize Stream dieksekusi. Pemanggilan ini akan terjadi sampai semua data yang dibutuhkan tersedia. Jika ISpSREngineSite::Read menghasilkan data yang lebih sedikit dari yang dibutuhkan, yang berarti tidak ada data lagi, maka *engine* akan menghentikan eksekusi ISpSREngine::RecognizeStream.

DDI memungkinkan *engine* untuk hanya mempunyai satu buah thread yang dieksekusi antara SAPI dan *engine*. Satu-satunya metode yang tidak mengijinkan Isp SREngine untuk masuk dan keluar secara cepat ialah ISpSREngine::Recognize Stream. *Engine* akan selalu tereksekusi dan terpisah di dalam ::RecognizeStream sampai terjadi kegagalan atau tidak ada lagi data yang diterima dari ISpSREngine::Read. Ketika *engine* mempunyai kesempatan untuk memberikan SAPI giliran melakukan pemanggilan kembali ke ISpSREngine, maka *engine* harus memanggil Isp SREngine::Synchronize dan memberikan indeks posisi *stream* dimana *engine* telah selesai melakukan pengenalan data. Dari manapun Synchronize dipanggil, SAPI dapat melakukan pemanggilan kembali ke metode ISpSREngine yang lain kecuali Recognize. Sebagai contoh, *speaker* dapat berubah, *grammars* dapat di-*unload*, diaktifkan ataupun dinon-aktifkan secara dinamis. Pada bagian manapun, Isp SREngine akan dipanggil hanya pada *thread* semula yang membuat ISpSREngine atau thread dimana *engine* memanggil Isp SREngineSite::Synchronize. Jika *engine* hanya memanggil ISpSREngineSite::Synchronize pada *thread* yang sama dengan ISpSREngine::RecognizeStream, maka hal ini berarti SAPI hanya memanggil satu *thread* yang merupakan turunan dari ISpSREngine.

4. IMPLEMENTASI PROGRAM

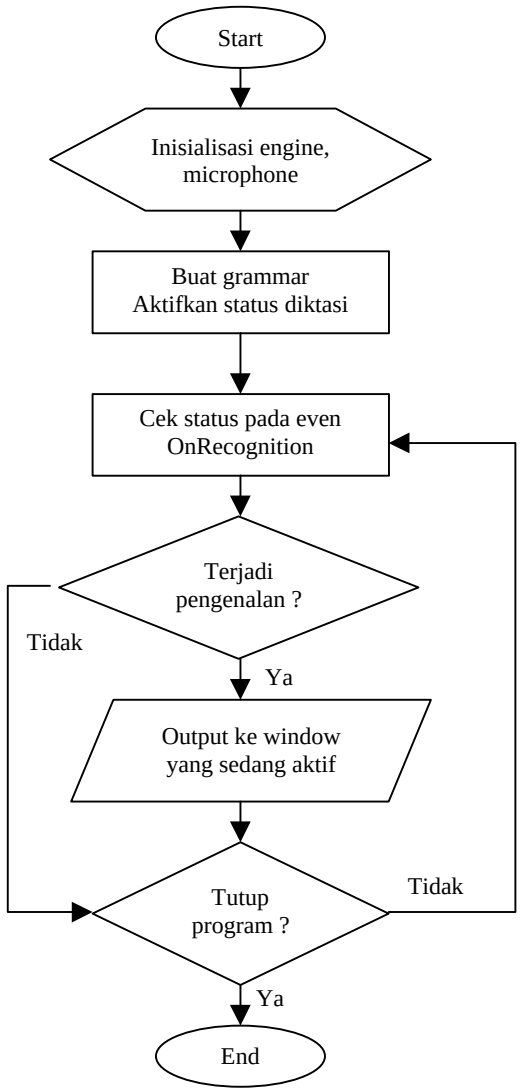
SAPI 5.1 memberikan hampir semua antar muka, tipe dan konstanta yang penting dengan menggunakan registered type library. Hal ini menyebabkan fungsi SAPI

5.1 dapat diakses melalui *late bound* ataupun *early bound automation*. Pada program ini digunakan *early bound automation*. Untuk dapat menggunakan *early bound automation* ini, maka terlebih dahulu harus meng-*import type library* dengan cara memilih menu Project – Import Type Library pada Delphi, sehingga akan tampil daftar ActiveX yang berada pada sistem. *Type library* untuk SAPI 5.1 ini ialah Microsoft Speech Object Library (Version 5.1). Pada saat hendak melakukan instalasi *type library* ini, juga harus menyertakan component wrapper classes untuk tiap obyek *automation* yang dilakukan dengan cara melakukan pemilihan pada pilihan Generate Component Wrapper.

Hasil dari instalasi ini ialah file SpeechLib_TLB.pas yang kemudian dapat di-instalasi sebagai package pada Delphi. Setelah selesai proses instalasi package, maka akan didapat sebanyak 19 komponen baru yang merupakan komponen SAPI 5.1. Tiap komponen ini merupakan implementasi dari antar muka dan mempunyai nama mirip dengan nama antar muka tersebut, misal : komponen TSpSharedRecognizer merupakan implementasi antar muka SpShared Recognizer.

Pada pembuatan sistem, terdapat dua pilihan untuk menggunakan *engine*, yaitu *shared recognizer* (TSpSharedRecognizer) atau InProc *recognizer* (TspInproc Recognizer). Dengan menggunakan TspInproc Recognizer akan didapat hasil yang lebih efisien karena TSpInprocRecognizer akan menempati memori proses aplikasi, tetapi mempunyai kelemahan yaitu tidak bisa berbagi *resource* dengan aplikasi lain. Sebagai contoh adalah aplikasi lain tidak dapat menerima masukan dari *microphone* sampai aplikasi pertama ditutup. Dengan menggunakan *shared recognizer*, aplikasi dapat saling berbagi *resource*.

Recognizer menggunakan *recognition context* untuk mengidentifikasikan kapan aplikasi akan aktif. Hal ini diimplementasikan dengan SpInprocRecoContext atau SpSharedRecoContext. Aplikasi dapat menggunakan hanya satu *context* ataupun menggunakan beberapa *context*. *Recognition context* ini memungkinkan untuk memulai atau mengakhiri pengenalan, mengatur *grammar* dan menerima notifikasi penting.



Gambar 2. Blok Diagram Sistem.

Pada pengenalan pembicaraan terdapat bagian proses yang menentukan jenis kata yang diucapkan. Proses ini menggunakan *grammar* untuk menentukan kata tersebut jika *grammar* tersedia. SAPI 5.1 mendefinisikan format untuk *grammar* dalam bentuk XML. Pada diktasi, *grammar* digunakan untuk mengidentifikasi beberapa kata khusus yang sering digunakan. *Grammar* tidak dapat digunakan untuk menganalisa keseluruhan kata pada diktasi karena tidak mungkin untuk mendaftar seluruh kata yang dipergunakan dalam percakapan ke dalam *grammar*. Biasanya sistem akan menggunakan metode tertentu serta *context analysis* untuk melakukan pengenalan kata pada diktasi. Dan hasil

proses ini akan lebih akurat dengan menggunakan tambahan *grammar* yang telah didefinisikan.

Untuk memulai pembuatan aplikasi, maka digunakan komponen TspSharedReco Context. Sedangkan TSpSharedRecognizer akan dibuat secara otomatis oleh *recognizer*, sehingga komponen tersebut tidak perlu digunakan secara eksplisit, kecuali jika hendak menggunakan metode yang terdapat pada komponen tersebut secara langsung. Langkah berikutnya adalah mendefinisikan *grammar* yang akan digunakan pada aplikasi dengan menuliskan kode berikut pada saat aplikasi pertama kali dijalankan :

```
SpSharedRecoContext1.EventInterests := SREAllEvents;
MyGrammar := SpSharedRecoContext1.CreateGrammar(0);
MyGrammar.DictationSetState(SGDSActive);
```

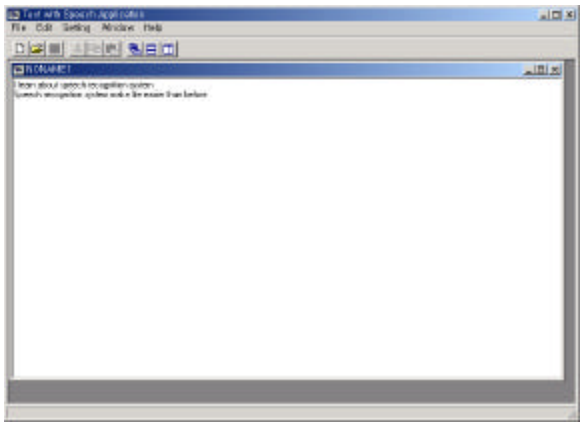
Pada saat terjadi notifikasi pada aplikasi seperti permulaan suara, akhir suara, permulaan frasa, akhir frasa dan lain-lain, maka akan terjadi pemanggilan pada standard even Delphi, dimana pada even tersebut bisa ditambahkan kode yang diperlukan seperti penampilan status. Even yang paling penting adalah OnRecognition, dimana even ini terjadi setelah sistem selesai melakukan pengenalan pada sebuah kata. Hasil dari OnRecognition ini akan dikirimkan melalui sebuah parameter yang berupa obyek ISpeechRecoResult. Untuk mengambil hasil dari pengenalan dapat dilihat pada kode berikut :

```
procedure TMainForm.SpSharedRecoContext1Recognition(Sender: TObject;
  StreamNumber: Integer; StreamPosition: OleVariant;
  RecognitionType: TOleEnum; var Result: OleVariant);
var
  RecogText: ISpeechRecoResult;
  FMDIChild: TMDIChild;
begin
  RecogText := IDispatch(Result) as ISpeechRecoResult;
  FMDIChild := MainForm.ActiveMDIChild as TMDIChild;
  if FMDIChild <> nil then
    FMDIChild.Memo1.SelText := RecogText.PhraseInfo.GetText(0, -1, True) + ' ';
end;
```

Hasil dari pengenalan pembicaraan ini diimplementasikan untuk menambahkan kata pada TMemo, dimana TMemo tersebut lebih lanjut dapat disimpan sebagai file teks.

5. PENGUJIAN

Berikut disajikan tampilan aplikasi beserta hasil pengujian pengenalan pembicaraan yang telah dilakukan.



Gambar 3. Tampilan Aplikasi.

Pengujian dilakukan oleh dua orang yang berbeda dengan melakukan diktasi dua paragraf artikel yang mempunyai jumlah kata berbeda, masing-masing diulangi sebanyak tiga kali dan kemudian dihitung jumlah kata yang dikenali secara benar oleh sistem untuk menentukan persentase keakuratan pengenalan pembicaraan.

Tabel 1. Pengujian sistem

Orang/ Paragraf	Percobaan ke	Jumlah kata	Pengenalan kata yang salah	Pengenalan kata yang benar	% keakuratan
1/1	1	148	28	120	81.08
1/1	2	148	26	122	82.43
1/1	3	148	20	128	86.49
1/2	1	255	50	205	80.39
1/2	2	255	40	215	84.31
1/2	3	255	44	211	82.75
2/1	1	148	20	128	86.49
2/1	2	148	15	133	89.86
2/1	3	148	16	132	89.19
2/2	1	255	48	207	81.18
2/2	2	255	47	208	81.57
2/2	3	255	41	214	83.92

Dari tabel pengujian sistem di atas terlihat bahwa rata-rata persentase keakuratan sistem adalah sekitar 85 %, dan

sistem akan menjadi semakin akurat setelah dilakukan beberapa kali percobaan yang sama. Jadi dengan semakin sering sistem digunakan, maka keakuratan akan menjadi semakin tinggi.

6. KESIMPULAN DAN SARAN

- Dengan menggunakan Microsoft Speech Engine pembuat aplikasi dapat meng-implementasikan kemampuan pengenalan pembicaraan yang dibuat untuk sistem operasi Windows secara cepat dan mudah dan tidak tergantung bahasa pemrograman yang dipakai.
- Dengan menggunakan SAPI 5.1 untuk pengenalan pembicaraan, aplikasi tidak terbatas menggunakan salah satu *engine* tertentu saja untuk pengenalan pembicaraan, tetapi dapat menggunakan *engine* lain yang diinginkan selama *engine* tersebut didesain sesuai dengan standard SAPI 5.1.
- SAPI 5.1 memberikan hampir semua antar muka, tipe dan konstanta yang penting melalui registered type library, sehingga memungkinkan pembuat aplikasi untuk mengakses SAPI 5.1 melalui *late bound* ataupun *early bound automation* secara mudah.
- Untuk pengembangan lebih lanjut bisa dilakukan perbandingan kemampuan *engine* yang mendukung SAPI 5.1 yang ada di pasaran sehingga didapatkan *engine* yang mempunyai keakuratan tinggi.

DAFTAR PUSTAKA

1. Amundsen, Michael C. *MAPI, SAPI, and TAPI Developer's Guide*. Indianapolis : Sams Publishing. 1996.

2. Long, Brian. *Speech Synthesis & Speech Recognition*. http://www.blong.com/Conferences/DCon2002/Speech/Speech.htm

3. Microsoft Website. http://microsoft.com/speech/techinfo/apioverview/

4. Microsoft Speech SDK 5.1