PERANCANGAN APLIKASI *VOICE USER INTERFACE* DENGAN MENGGUNAKAN MICROSOFT SPEECH API

Oleh :
Budhi Supriyono (L2F 099 583)
Jurusan Teknik Elektro Fakultas Teknik
Universitas Diponegoro Semarang

Abstrak *- Seiring dengan kemajuan teknologi, cara komunikasi antara manusia dan komputer telah mengalami perubahan dari waktu ke waktu. Perubahan tersebut dengan tujuan supaya komunikasi antara manusia dan komputer dapat berlangsung secara lebih efisien. Metode komunikasi yang cukup reliable dan user friendly yang masih digunakan sampai sekarang adalah GUI (Graphical User Interface).*

Metode baru yang baru diperkenalkan untuk menggantikan GUI adalah VUI (Voice User Interface). Perkembangan VUI ini ditunjang oleh teknologi ASR (Automatic Speech Recognizer). Dengan adanya teknologi ASR tersebut memungkinkan komputer untuk mengenali suara manusia sesuai dengan bahasa yang digunakan. Dengan VUI ini nantinya diharapkan komunikasi antara manusia dan komputer dapat dilakukan se-natural mungkin seperti komunikasi antara manusia dan manusia.

Microsoft menyertakan teknologi ASR tersebut pada Sistem Operasi yang dibuatnya. Oleh Microsoft, teknologi ASR digabungkan dengan teknologi Speech Synthesis, yaitu teknologi yang dapat mensintesis suara manusia, menjadi suatu API (Application Programming Interface) yang disebut Microsoft Speech API.

Kata Kunci : Speech API, Voice User Interface

I. PENDAHULUAN

1.1 Latar Belakang

Teknologi ASR (*Automatic Speech Recognition*) memungkinkan komputer untuk mengenali suara manusia, walaupun masih terbatas pada bahasa tertentu saja. Teknologi ini secara perlahan-lahan telah mengubah cara berkomunikasi antara manusia dan komputer. Komunikasi antara manusia dan komputer dengan menggunakan suara belum begitu banyak digunakan saat ini.

VUI (*Voice User Interface*) merupakan pengembangan dari teknologi ASR tersebut. VUI ini perlahan-lahan menjadi metode alternatif komunikasi antara manusia dan komputer selain GUI (*Graphical User Interface*) yang sudah bertahun-tahun digunakan. Secara garis besar, VUI adalah metode komunikasi antara manusia dan komputer dengan menggunakan media suara. Dengan metode VUI ini diharapkan komunikasi antara manusia dan komputer dapat berlangsung secara lebih efisien.

Berbagai aplikasi dapat dikembangkan berdasarkan metode VUI ini. Sebagai tahap awal, disusun rancangan aplikasi VUI yang menggabungkan antara metode GUI dan VUI. Dengan disusunnya rancangan ini, nantinya dapat dikembangkan untuk aplikasi VUI lainnya.

1.2 Tujuan

Tujuan dari tugas akhir ini adalah merancang desain dan struktur dari Aplikasi VUI (*Voice User Interface*) pada Sistem Operasi Windows dengan menggunakan Microsoft *Speech API*.

1.3 Pembatasan Masalah

Pada Tugas Akhir ini pembahasan akan dibatasi pada hal-hal berikut ini :

1. Tugas Akhir ini dirancang dengan menggunakan *engine* dari Microsoft Speech API 5.1.
2. Tugas Akhir ini dirancang menggunakan metode perancangan OOAD (*Object-Oriented Analysis*

and Design) dan mengunakan bahasa pemodelan UML (*Unified Modeling Language*).

3. Pada Tugas Akhir ini tidak membahas proses pengenalan suara (pengenalan pola) dan proses sintesis suara.
4. Pada Tugas Akhir ini tidak membahas algoritma *Artificial Intelligence* yang digunakan oleh Microsoft Speech API 5.1.
5. Pada Tugas Akhir ini tidak membahas struktur dan jenis database yang digunakan oleh Microsoft Speech API 5.1.

Tugas Akhir ini dirancang untuk berjalan di atas sistem operasi Microsoft Windows 2000 ke atas.

II. PRINSIP KERJA APLIKASI *VOICE USER INTERFACE*

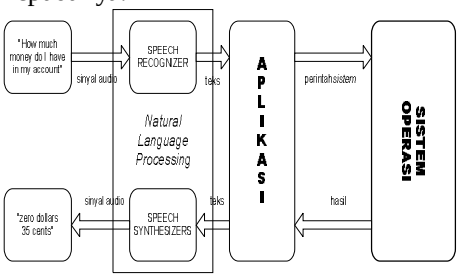
2.1 Pengertian VUI

Voice User Interface (VUI) adalah suatu metode komunikasi antara komputer dengan manusia dengan menggunakan suara[6]. VUI ini merupakan pengembangan dari *Graphical User Interface* (GUI) yang menggunakan visualisasi gambar atau tulisan untuk berkomunikasi antara komputer dengan manusia.

2.2 Prinsip Kerja Aplikasi VUI

Prinsip kerja secara umum dari Aplikasi *Voice User Interface* adalah Aplikasi VUI ini akan mendengarkan perintah dari *user* dan kemudian memberikan respon sesuai dengan perintah yang diberikan. Proses pengenalan suara *user* dilakukan oleh *Speech Recognizer* yang berada di dalam Speech API, kemudian *Speech Recognizer* memberikan kata atau kalimat yang diucapkan *user* kepada Aplikasi VUI dalam bentuk teks. Teks tersebutlah yang diproses oleh Aplikasi VUI untuk menentukan respon apa yang akan dilakukan atas perintah yang diberikan oleh *user*.

Aplikasi VUI ini juga dapat memberikan pesan atau respon berupa suara, hal ini dilakukan Aplikasi VUI dengan cara memberikan kata atau kalimat yang akan disuarakan kepada *Speech Synthesizer* yang berada di dalam Speech API. Proses selanjutnya *Speech Synthesizer* akan mensintesis suara sesuai dengan kata atau kalimat yang diberikan kepadanya.



Gambar 2.1 Diagram Prinsip Kerja Aplikasi VUI

2.3 Teknologi-teknologi yang mendasari Aplikasi VUI

Aplikasi *Voice User Interface* bukanlah aplikasi yang berdiri sendiri. Aplikasi VUI tersusun atas teknologi-teknologi lainnya yang memungkinkan aplikasi ini berjalan sebagaimana mestinya. Teknologi-teknologi yang mendasari Aplikasi VUI ini adalah :

1. *Natural Language Processing* (NLP).
2. *Automatic Speech Recognition* (ASR).
3. *Speech Synthesis.*
4. *Object Linking and Embedding* (OLE).
5. Windows *Application Programming Interface* (API).

2.3.1 Natural Language Processing

Teknologi *Natural Language Processing* (NLP) dikembangkan dengan tujuan supaya diperoleh suatu teknologi yang dapat menganalisa, mengerti dan menghasilkan suara (bahasa) yang digunakan oleh manusia se-*natural* mungkin. Sehingga diharapkan menusia dapat berkomunikasi dengan komputer seperti manusia berkomunikasi dengan manusia.

Teknologi ini telah dikembangkan selama tiga dasawarsa dan baru beberapa tahun ini teknologi NLP tersebut telah mencapai tahap produksi. Pemrosesan suara (bahasa) yang dapat dengan mudah dilakukan oleh manusia ternyata sangat sulit dilakukan oleh komputer. Hal ini disebabkan oleh sifat dasar dari komputer, yaitu komputer hanya dapat menghitung

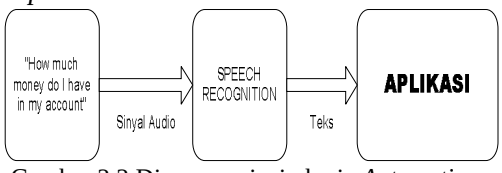
nilai-nilai yang konsisten, sedangkan bahasa bersifat ambigu. Keambiguan tersebut yang menyebabkan suatu kata dapat berarti lebih dari satu.

2.3.2 Automatic Speech Recognition

Automatic Speech Recognition (ASR) merupakan teknologi yang dikembangkan dari teknologi NLP. *Automatic Speech Recognition* adalah teknologi yang dapat mengenali suara manusia dalam bahasa tertentu dan merubah suara yang dikenalinya kedalam bentuk teks[5,7,8].

Secara umum prinsip kerja dari ASR adalah ketika seseorang berbicara kepada komputer, program menangkap suara orang tersebut melalui *microphone* dan mengubahnya menjadi sinyal digital. Kemudian program menganalisa sinyal digital tersebut dengan membandingkannya dengan *digital pattern* yang ada dalam databasenya. Setelah itu akan diambil *digital pattern* yang paling besar prosentase kemiripannya, kemudian dari *digital pattern* tersebut diubah menjadi teks. Karena setiap manusia memiliki karakteristik suara yang berbeda-beda, maka diberikan suatu metode untuk melatih program dan kemudian data-data spesifik tentang karakter suara tersebut disimpan dalam database dengan tujuan supaya proses pengenalan suara berikutnya memiliki prosentase keberhasilan yang lebih besar.

Proses pengenalan suara ini sangat bergantung pada bahasa yang digunakan, karena setiap bahasa memiliki cara pengucapan yang berbeda. Sehingga teknologi ASR ini bersifat *language dependent*[8].

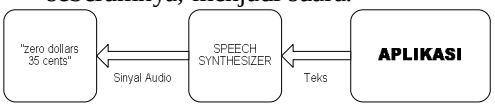


Gambar 2.2 Diagram prinsip kerja *Automatic Speech Recognition*

2.3.3 Speech Synthesis

Speech Synthesis juga merupakan pengembangan dari teknologi NLP. *Speech Synthesis* adalah teknologi yang mampu membuat sintesis suara dalam bahasa tertentu berdasarkan teks yang diberikan kepadanya[5,7]. Prinsip kerja dari *Speech Synthesis* ini berkebalikan dengan prinsip kerja dari ASR. Kalau ASR berusaha untuk mengenali suara maka *Speech Synthesis* ini berusaha untuk menghasilkan suara se-*natural* mungkin sesuai dengan input yang diberikan kepadanya (biasanya berupa teks). Secara umum *Speech Synthesis* terdiri dari dua sub-sistem, yaitu :

1. *Text to Phoneme*, berfungsi untuk mengubah masukan (berupa teks) dalam bahasa tertentu menjadi rangkaian kode-kode bunyi yang direpresentasikan sebagai kode fonem, durasi dan *pitch*-nya. Bagian ini sangat tergantung dari bahasa yang digunakan atau *language dependant*.
2. *Phoneme to Voice*, berfungsi untuk mengubah kode-kode fonem, durasi dan *pitch* ,dari sub-sistem sebelumnya, menjadi suara.



Gambar 2.3 Diagram prinsip kerja *Speech Synthesizer*

2.3.4 Object Linking and Embedding

Object Linking and Embedding (OLE) merupakan pengembangan dari teknologi Dynamic Data Exchange (DDE). *Object Linking and Embedding* adalah teknologi yang dikembangkan oleh Microsoft untuk mendukung pertukaran informasi antar aplikasi, dimana informasi yang dikirimkan masih dapat dimodifikasi[15]. Konsep dasarnya adalah informasi direpresentasikan sebagai suatu objek yang menghubungkan (link) antar aplikasi atau keseluruhan dokumen objek tersebut di-*embedded* ke aplikasi tujuan. Dari sinilah muncul istilah *Object Linking and Embedding*.

Otomatisasi adalah sebagian aspek dari OLE yang memungkinkan sebuah aplikasi untuk mengontrol, atau mengotomatisasi aplikasi lainnya. Aplikasi yang akan dikontrol disebut *Automation Server*, sedangkan aplikasi

yang melakukan kontrol disebut *Automation Controller* atau *Automation Client*. Otomatisasi melalui OLE ini memungkinkan informasi dipertukarkan antara *Automation Server* dan *Automation Client* dan juga memungkinkan *Automation Client* untuk memerintahkan suatu fungsi dijalankan oleh *Automation Server*. Oleh karena otomatisasi dijalankan dengan menggunakan objek, maka aplikasi lainnya dapat membaca dan menulis *properties* dari objek tersebut dan dapat pula menjalankan *method* yang dimiliki oleh objek tersebut.

Suatu aplikasi yang ingin berfungsi sebagai *Automation Server* haruslah teregistrasi terlebih dahulu, yang berarti aplikasi tersebut harus memiliki informasi di dalam *registry* windows untuk mendeskripsikan dirinya. Sehingga apabila nanti ada suatu aplikasi *client* yang ingin menggunakan dirinya maka Windows dapat menunjukkan dimana lokasi dari *Automation Server* tersebut.

Ada tiga cara yang dapat digunakan sebuah *Automation Client* untuk mengontrol *Automation Server*. Cara-cara tersebut adalah :

1. Mengontrol *Automation Server* dengan menggunakan *Variants*.
2. Mengontrol *Automation Server* dengan menggunakan *Interfaces*.
3. Mengontrol *Automation Server* dengan menggunakan *Components*.

2.3.5 Windows Application Programming Interface (API)

Windows *Application Programming Interface* (API) adalah sekumpulan fungsi dan konstanta yang terdapat dalam *file-file* DLL yang menyusun Sistem Operasi Windows[14]. Karena terdapat dalam *file library* (DLL) maka fungsi-fungsi dan konstanta-konstanta tersebut dapat digunakan oleh aplikasi atau *library* lainnya. Windows API biasanya disusun dalam bahasa pemrograman C/C++. Bahasa C dipilih karena kecepatan dan kedekatannya dengan *hardware*.

Windows menyediakan banyak sekali API yang memiliki fungsi berbeda-beda. Berikut ini adalah beberapa API yang disediakan oleh Sistem Operasi Windows :

1. Win32 API, API yang disediakan Windows untuk melakukan pemrograman Windows secara umum. Fungsi-fungsi yang terdapat dalam Win32 API ini sangat lengkap mulai dari menu, button, window, eksekusi program dan lain-lain.
2. *Telephony* API (TAPI), API yang digunakan untuk melakukan fungsi-fungsi *telephony* dengan menggunakan Sistem Operasi Windows.
3. *Messaging* API (MAPI), API yang digunakan untuk melakukan pengiriman pesan (termasuk pengiriman *email*) dengan menggunakan Sistem Operasi Windows.
4. *Speech* API (SAPI), API yang disediakan Windows untuk menambahkan fungsi *speech* ke dalam aplikasi yang berjalan di atas Sistem Operasi Windows.
5. *Cryptography* API, API untuk melakukan fungsi-fungsi enkripsi dan dekripsi. Banyak sekali metode enkripsi/dekripsi yang disediakan oleh *cryptography* API ini.
6. dan masih banyak API lainnya (termasuk yang tidak didokumentasikan oleh Microsoft).

Seperti telah disinggung sebelumnya bahwa API sebenarnya hanyalah sebuah fungsi. Seperti halnya fungsi-fungsi lainnya, fungsi tersebut dapat memiliki *parameter-parameter* dan dapat mengembalikan nilai. Bedanya hanyalah fungsi-fungsi tersebut ditulis dalam bahasa C/C++ sehingga bagi yang belum familiar dengan bahasa C/C++ biasanya akan mengalami kesulitan.

2.4 Unified Modeling Language (UML)

Unified Modelling Language (UML) adalah sebuah bahasa untuk menentukan, visualisasi, konstruksi dan mendokumentasi informasi-informasi dari sistem *aplikasi*, untuk memodelkan bisnis dan sistem non-*aplikasi* lainnya. UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam

memodelkan sistem yang besar dan kompleks.

Untuk membuat suatu model, UML memiliki diagram grafis sebagai berikut :

1) *Use-case diagram*
2) *Behavior diagram :*
   a) *Statechart diagram*
   b) *Activity diagram*
   c) *Interaction diagram :*
      i) *Sequence diagram*
      ii) *Collaboration diagram*
3) *Class diagram*
4) *Implementation diagram :*
   a) *Component diagram*
   b) *Deployment diagram*

Diagram-diagram tersebut diberi nama berdasarkan sudut pandang yang berbeda-beda terhadap sistem dalam proses analisis atau rekayasa. Dibuatnya berbagai jenis diagram di atas karena :

1. setiap sistem yang kompleks selalu paling baik jika didekati melalui himpunan berbagai sudut pandang yang kecil, yang satu sama lain hampir saling bebas (*independent*). Sudut pandang tunggal senantiasa tidak mencukupi untuk melihat sistem yang besar dan kompleks.
2. Diagram yang berbeda-beda tersebut dapat menyatakan tingkatan yang berbeda-beda dalam proses rekayasa.
3. Diagram-diagram tersebut dibuat agar model yang dibuat semakin mendekati realitas.

Tujuan utama UML diantaranya untuk :

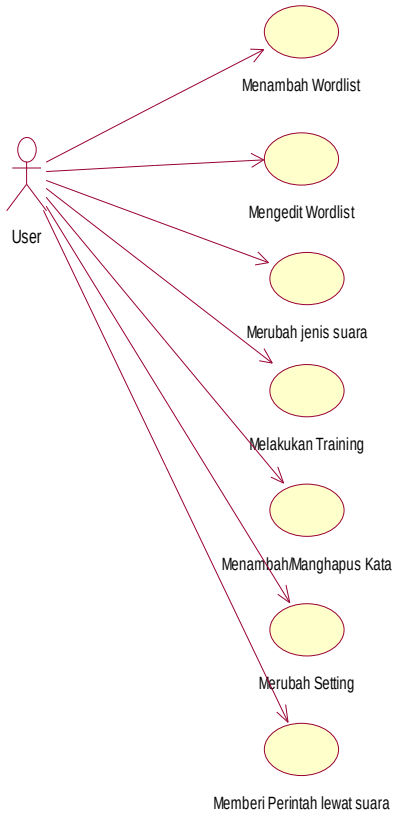
1. Memberikan model yang siap pakai, bahasa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
2. Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.
3. Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan.

III. PERANCANGAN APLIKASI DENGAN UML

UML adalah sebuah bahasa untuk menentukan, visualisasi, konstruksi dan mendokumentasi informasi-informasi dari sistem aplikasi, untuk memodelkan bisnis dan sistem non-aplikasi lainnya. Bahasa UML ini sangat erat kaitannya dengan *Object Oriented Analysis and Design* (OOAD), karena UML sendiri adalah bahasa pemodelan yang berbasiskan *Object Oriented* tapi bersifat *language independent*. Sehingga UML dapat digunakan oleh bahasa pemrograman apapun yang berbasiskan *Object Oriented*.

Use Case Diagram

Pada aplikasi *Voice User Interface* (VUI) yang berperan sebagai aktor adalah *user* atau pengguna aplikasi tersebut. User dapat menambahkan kata dalam *wordlist*, merubah *setting* dari *Speech* API, memilih jenis suara yang digunakan dan yang paling utama *user* dapat memberikan perintah melalui suara. Berikut ini adalah diagram *use case* dari aplikasi VUI.

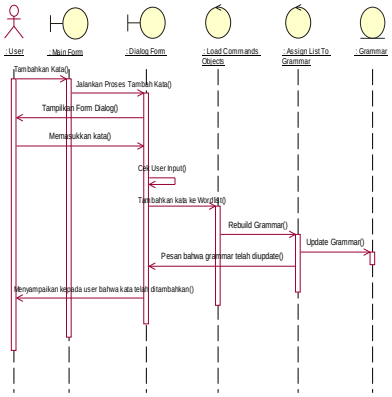


Gambar 3.1 Diagram *Use Case* aplikasi VUI

Pada gambar 3.1 menunjukkan diagram *use case* dengan aktor *user*. Berdasarkan gambar 3.1 menunjukkan fungsi yang harus dipenuhi oleh sistem agar aplikasi VUI dapat berjalan dengan semestinya.

3.1.1 Menambah *Wordlist*

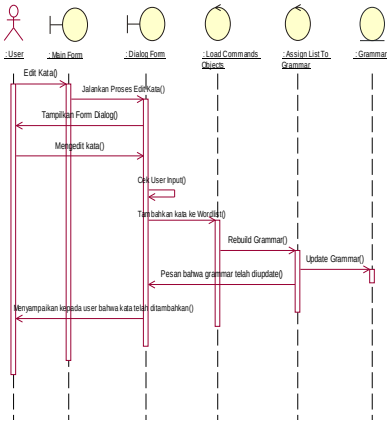
Fungsi ini digunakan oleh *user* untuk menambahkan kata/kalimat ke dalam *wordlist* (daftar kata/kalimat yang akan didengarkan oleh *engine recognizer*). Setelah menambahkan kata/kalimat ke dalam *wordlist*, kemudian fungsi ini akan meng-*update grammar* dari *engine*.



Gambar 3.2 *Sequence Diagram* yang menjelaskan *use case* Menambah wordlist

3.1.2 Mengedit *Wordlist*

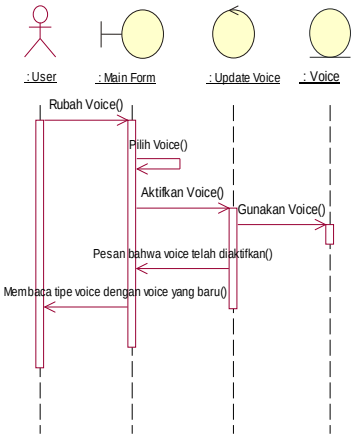
Fungsi ini digunakan oleh *user* untuk mengubah kata/kalimat yang sudah ada dalam *wordlist*. Setelah mengubah kata/kalimat dalam *wordlist*, kemudian fungsi ini akan meng-*update grammar* dari *engine*.



Gambar 3.3 *Sequence diagram* yang menjelaskan *use case* Mengedit *Wordlist*

3.1.3 Merubah jenis suara

Fungsi ini digunakan untuk memilih dan mengubah jenis *voice*/suara yang digunakan oleh *speech synthesizer engine*. Karena ada beberapa jenis suara yang disediakan oleh Microsoft *Speech* API yang dapat digunakan, sehingga *user* dapat memilih jenis suara yang dia senangi.

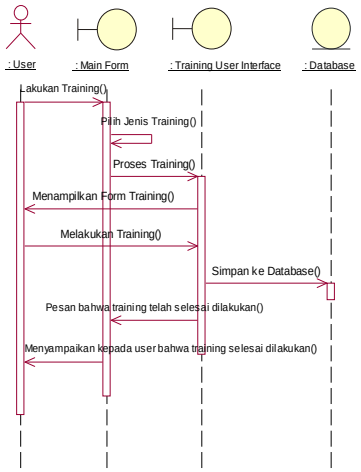


Gambar 3.4 *Sequence diagram* yang menjelaskan *use case* Merubah jenis suara.

3.1.4 Melakukan Training

Fungsi ini digunakan oleh *user* untuk melatih *speech recognizer engine* supaya dapat lebih baik lagi dalam mengenali suara dari *user*. Fungsi ini diperlukan karena setiap suara manusia memiliki karakter yang berbeda-beda, sehingga dengan fungsi training ini

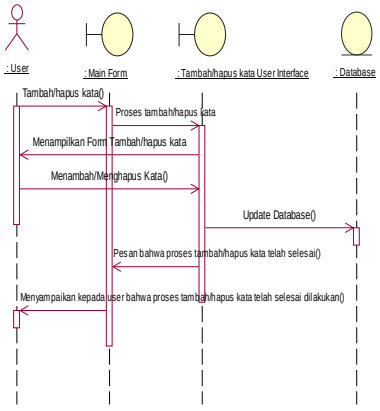
diharapkan *speech recognizer engine* akan mengenali suara *user* dengan lebih baik.



Gambar 3.5 *Sequence diagram* yang menjelaskan *use case* Melakukan Training

3.1.5 **Menambah/Menghapus kata**

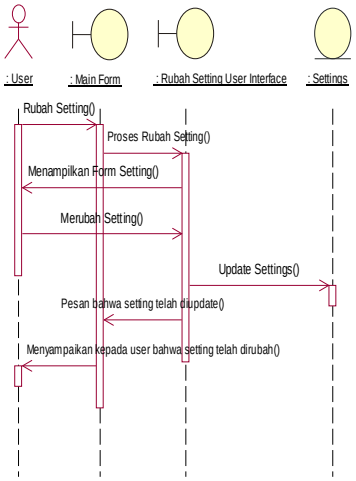
Fungsi ini digunakan untuk menambah/menghapus kata ke/dari *database engine*. *Database engine* telah berisi seluruh kata yang mungkin dalam bahasa inggris, tapi seiring dengan perkembangan budaya dimungkin adanya penambahan kata-kata baru atau pengucapan-pengucapan baru. Sehingga fungsi ini memberikan kemampuan kepada *user* untuk menambahkan kata-kata yang belum ada dalam *database engine.*



Gambar 3.6 *Sequence diagram* yang menjelaskan *use case* Menambah/menghapus kata

3.1.6 **Merubah *Setting***

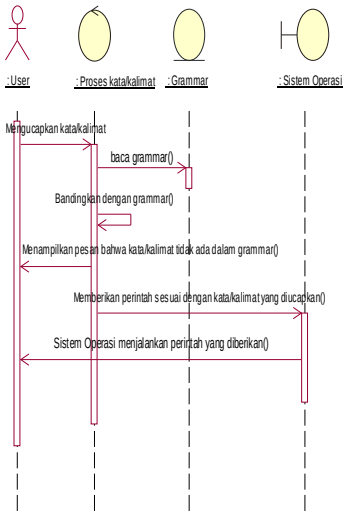
Fungsi ini memberikan kemampuan kepada *user* untuk merubah *setting* dari *speech engine* (baik itu *speech recognizer engine* maupun *speech synthesizer engine*). *Setting* yang dapat dirubah oleh *user* antara lain : perbandingan antara kepekaan dan kecepatan pengenalan suara, volume *speech synthesizer engine* dan merubah *pitch* dari *speech synthesizer engine*.



Gambar 3.7 *Sequence diagram* yang menjelaskan *use case* Merubah Setting

3.1.7 **Memberi perintah lewat suara**

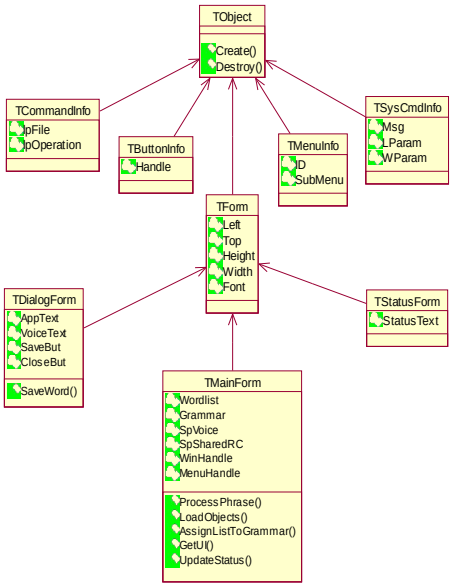
Dalam aplikasi VUI interaksi antara *user* dan aplikasi sebagian besar dilakukan dengan suara. Dalam fungsi ini *user memberikan* perintah kepada aplikasi melalui suara dan aplikasi akan melakukan perintah *user* tersebut berdasarkan *grammar* dan *wordlist*.



Gambar 3.8 *Sequence diagram* yang menjelaskan *use case* Memberi perintah lewat suara

Class Diagram

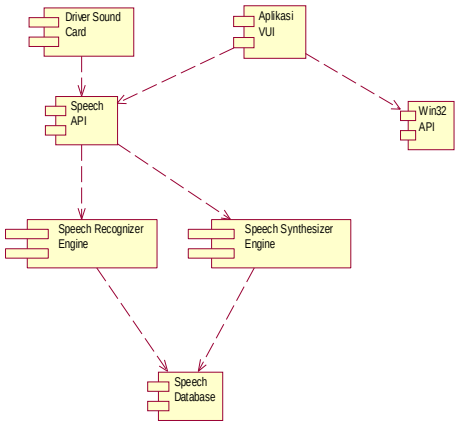
Class Diagram digunakan untuk memvisualisasikan hubungan antar kelas dan penjelasan detail tiap-tiap kelas dalam model desain. Model desain menggambarkan objek-objek dan hubungan antar objek-objek tersebut dalam abstraksi yang mendekati *source code*. Tujuannya adalah untuk memudahkan untuk melakukan *coding* dari sistem yang dibuat.



Gambar 3.9 *Class Diagram* dari sistem

Component Diagram

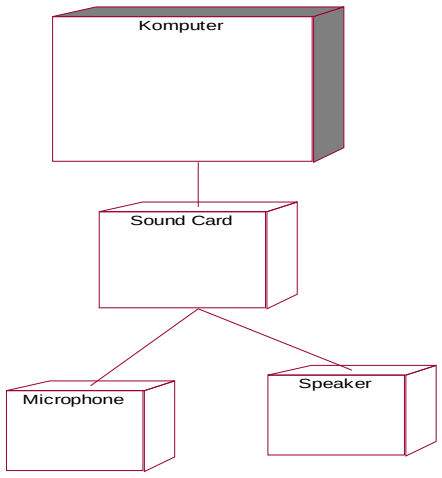
Component Diagram menggambarkan alokasi semua kelas dan objek kedalam komponen-komponen dalam desain fisik sistem aplikasi. Diagram ini memperlihatkan pengaturan dan kebergantungan antara komponen-komponen aplikasi, seperti kode sumber, kode biner dan komponen tereksekusi (*executable components*).



Gambar 3.10 *Component Diagram* dari desain aplikasi VUI

Deployment Diagram

Setiap model hanya memiliki satu *deployment diagram*. Diagram ini memperlihatkan pemetaan aplikasi kepada *hardware*. Berikut ini adalah *deployment diagram* dari desain aplikasi VUI :



Gambar 3.11 *Deployment Diagram* dari desain aplikasi VUI

IV. IMPLEMENTASI DESAIN

Berdasarkan desain aplikasi yang telah dibuat akan diimplementasikan menjadi kode-kode yang dapat di-*compile* menjadi *file executable* dengan menggunakan bahasa pemrograman Delphi. Fungsi-fungsi yang digunakan dalam aplikasi VUI ini adalah fungsi-fungsi yang terdapat dalam Win32 API dan *Speech* API. Desain tampilan dilakukan dengan program Delphi 7 Enterprise demikian pula dengan pembuatan kode sumber-nya.

4.1 Desain Form

Pada perancangan aplikasi VUI ini digunakan 4 desain form, masing-masing form tersebut adalah :

1. MainForm, merupakan form utama yang berguna untuk melakukan perubahan setting dan menambah/mengedit *wordlist*.
2. DialogForm, merupakan form dialog yang digunakan untuk menambah atau mengedit *Wordlist*.
3. ReadForm, merupakan form yang digunakan untuk menampilkan dokumen yang dibaca oleh aplikasi VUI.
4. DictationForm, merupakan form yang digunakan untuk melakukan *dictation*.

4.2 Class Diagram

Kelas merupakan definisi formal suatu objek. Kelas berfungsi sebagai *template* bagi objek yang merupakan *instance* dari suatu kelas. Sering kali orang menyamakan antara kelas dan objek, padahal antara kelas dan objek adalah tidak sama. Kelas merupakan suatu tipe data yang telah didefinisikan oleh *user*. Sebuah kelas dapat memiliki *property, method* dan *event* yang berbeda-beda. Sedangkan objek adalah *instance* dari kelas tersebut.

Dalam aplikasi VUI ini terdapat beberapa kelas yang telah didefinisikan untuk digunakan dalam proses *coding*. Pada umumnya setiap form direpresentasikan dengan sebuah kelas, sehingga pada aplikasi VUI ini terdapat lima kelas turunan dari Tform karena dalam aplikasi VUI ini terdapat 4 form tampilan. Selain empat kelas tersebut, didefinisikan pula kelas lainnya sebagai pendukung dari empat kelas utama tadi.

Kelas-kelas yang didefinisikan dalam aplikasi VUI ini adalah :

1. Kelas TMainForm
2. Kelas TDialogForm
3. Kelas TReadForm
4. Kelas TDictationForm
5. Kelas TCommandInfo
6. Kelas TSysCmdInfo
7. Kelas TButtonInfo
8. Kelas TMenuInfo

Setiap kelas tersebut memiliki *properties, method* dan *event*. *Properties* adalah sifat atau karakteristik yang dimiliki oleh suatu kelas. *Method* adalah operasi yang bisa dilakukan oleh kelas tersebut. Sedangkan *event* adalah kejadian-kejadian yang dapat terjadi pada kelas tersebut.

Tidak semua kelas memiliki *properties, method* dan *event*. Bisa saja suatu kelas hanya memiliki *properties* dan *method* saja atau bahkan bisa saja suatu kelas hanya memiliki *properties* saja, seperti pada kelas TcommandInfo, TsysCmdInfo, TbuttonInfo dan TmenuInfo. Keempat kelas tersebut hanya memiliki *properties* saja.

4.3 Konstanta-konstanta

Dalam pemrograman aplikasi *Voice User Interface* ini dideklarasikan beberapa konstanta yang bertujuan untuk mempermudah dalam melakukan pengkodean. Konstanta-konstanta tersebut mewakili nilai-nilai yang menunjukkan pengenal suatu variabel maupun pesan-pesan yang digunakan secara internal dalam aplikasi *Voice User Interface* ini.

4.4 Fungsi-fungsi

Selain membuat kelas-kelas yang menyusun aplikasi *Voice User Interface* dibuat pula fungsi-fungsi yang bertujuan memudahkan dalam pemrograman aplikasi ini. Dalam aplikasi ini dideklarasikan dua buah fungsi, kedua fungsi tersebut adalah :

1. **Fungsi MakeFormShapeOnBmp**, Fungsi ini digunakan untuk membuat bentuk suatu form sesuai

dengan bentuk suatu bitmap yang telah ditentukan. Prinsip kerja dari fungsi ini adalah me-*load* suatu bitmap dan kemudian menyamakan setiap pixel dalam bitmap dengan setiap pixel dalam form yang ingin dibentuk. Jika dalam bitmap ditemukan pixel-pixel transparan, maka pixel-pixel dalam form yang bersesuaian akan pula dibuat menjadi transparan sehingga akan membuat tampilan form menjadi lebih menarik.

2. **Fungsi PostKeyEx32**, Fungsi PostKeyEx32 ini digunakan untuk mensimulasikan penekanan suatu tombol keyboard beserta kombinasinya. Kombinasi yang dimaksud disini adalah kombinasi suatu tombol keyboard dengan tombol shift, control atau alt. Tombol-tombol yang ingin disimulasikan tersebut harus dirubah terlebih dahulu menjadi kode ASCII.

4.5 Inisialisasi dan Deinisialisasi

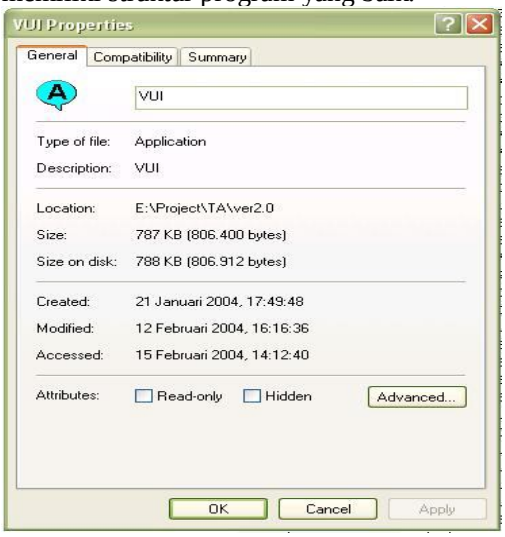
Aplikasi VUI ini memiliki fungsi inisialisasi dan deinisialisasi. Fungsi inisialisasi berguna untuk menginisialisasi variabel-variabel dan objek-objek yang digunakan oleh aplikasi VUI ini. Sedangkan fungsi deinisialisasi berguna untuk menghapus dari memori objek-objek yang digunakan oleh aplikasi VUI supaya tidak menjadi sampah dalam memori.

V. PENGUJIAN DAN PEMBAHASAN

Hal-hal yang diujikan terhadap aplikasi *Voice User Interface* (VUI) ini antara lain: ukuran file *executable*, penggunaan memori oleh aplikasi, pengaruh kondisi lingkungan terhadap jalannya aplikasi, pengaruh kualitas *microphone* pada aplikasi dan tanggapan responden terhadap aplikasi VUI ini. Hal-hal yang diujikan tersebut berguna untuk menentukan *parameter-parameter* pengukur keberhasilan perancangan aplikasi *Voice User Interface* dengan menggunakan Microsoft *Speech* API.

Ukuran *File*

Ukuran *file executable* dan jumlah memori yang digunakan merupakan *parameter* yang menunjukkan penggunaan *resource* oleh suatu aplikasi. Dengan penggunaan *resource* yang hemat akan mempercepat eksekusi suatu aplikasi, karena penggunaan *resource* yang hemat menunjukkan bahwa aplikasi tersebut didesain dengan bagus dan memiliki struktur program yang baik.



Gambar 5.1 Properties dari *file executable* Aplikasi *Voice User Interface*

Dari Gambar 5.1 terlihat bahwa *file executable* aplikasi *Voice User Interface* memiliki tipe aplikasi yang berlokasi di folder "E:\Project\TA\ver2.0", ukuran dari aplikasi *Voice User Interface* ini adalah 787 KB (806.400 bytes) dan besar media penyimpanan yang digunakan oleh aplikasi *Voice User Interface* ini adalah sebesar 788 KB (806.912 bytes).

Penggunaan Memory

Pengujian penggunaan *memory* oleh aplikasi *Voice User Interface* dilakukan dengan menggunakan aplikasi Task Manager yang merupakan aplikasi standard sistem operasi Windows untuk memonitor jalannya semua proses yang terjadi.

Pengujian penggunaan *memory* oleh Aplikasi *Voice User Interface* dilakukan pada tiga keadaan, yaitu :

1. Aplikasi sudah dijalankan tapi belum mendengarkan perintah dari *user*.
2. Aplikasi mendengarkan perintah dari *user* pada *mode command*.
3. Aplikasi mendengarkan perintah dari *user* pada *mode dictation.*

Tabel 5.1 manampilkan data hasil pengujian penggunaan *memory* oleh Aplikasi *Voice User Interface*.

Tabel 5.1 Hasil pengujian penggunaan *memory* oleh Aplikasi *Voice User Interface* dan *engine* Microsoft Speech API

Kondisi	Penggunaan *Memory*	
	Aplikasi *Voice User Interface*	*Engine* Microsoft Speech API
Belum mendengarkan perintah	5.524 KB	5.560 KB
Mode *Command*	5.928 KB	15.288 KB
Mode *Dictation*	5.852 KB	38.920 KB

Pengujian Pengaruh Lingkungan

Pengujian untuk mengetahui pengaruh kondisi lingkungan terhadap jalannya aplikasi *Voice User Interface* dilakukan dengan cara menjalankan aplikasi *Voice User Interface* pada tiga kondisi lingkungan yang berbeda. Ketiga kondisi lingkungan tersebut adalah :

1. Sepi/tenang, kondisi dimana tidak ada sumber suara lain selain suara dari *user*. Dalam pengujian, kondisi ini diwakili dengan kondisi dalam ruangan yang tertutup rapat pada waktu malam hari.
2. Berisik, kondisi dimana terdapat satu atau dua sumber suara lain selain suara dari *user*. Dalam pengujian, kondisi ini diwakili dengan kondisi dalam ruangan yang tertutup dimana dalam ruangan tersebut terdapat TV dan radio yang aktif.
3. Ramai, kondisi dimana terdapat banyak sumber suara lain selain suara dari *user*. Dalam pengujian, kondisi ini diwakili dengan kondisi di luar ruangan dan hanya berjarak ±5 meter dari jalan yang ramai.

Tabel 5.2 menampilkan hasil pengujian pengaruh lingkungan terhadap jalannya aplikasi *Voice User Interface*.

Tabel 5.2 Hasil pengujian pengaruh lingkungan terhadap aplikasi *Voice User Interface*

Kondisi Lingkungan	Prosentase ketepatan pengenalan suara	
	Mode Command	Mode Dictation
Sepi/tenang	100 %	92,5 %
Berisik	96 %	72,5 %
Ramai	80 %	47,5 %

Keterangan :

- Pengujian dilakukan dengan menggunakan Logitec Premium Headset
- Pengujian dilakukan setelah melakukan pelatihan pengenalan suara sebanyak tiga kali

Pengaruh kualitas *microphone*

Aplikasi *Voice User Interface* ini menerima *input* dari *user* dengan menggunakan *microphone*, sehingga sedikit banyak kualitas dari *microphone* yang digunakan akan mempengaruhi jalannya aplikasi *Voice User Interface* ini. Untuk mengetahui seberapa besar pengaruh kualitas *microphone* terhadap jalannya aplikasi *Voice User Interface* ini telah dilakukan pengujian dengan menggunakan berbagai jenis *microphone* yang memiliki kualitas berbeda-beda, jenis-jenis *microphone* yang digunakan dalam pengujian ini adalah sebagai berikut :

1. SONY Dynamic Microphone VJ-F22/C, *microphone* ini mewakili jenis *microphone* dengan kualitas terendah. *Microphone* jenis ini biasa digunakan untuk karaoke dengan *Tape Deck*.
2. SPEED SP-501 Headset, *microphone* ini mewakili jenis *microphone* dengan kualitas sedang. Headset ini dibuat khusus untuk digunakan pada komputer tapi tidak memiliki kemampuan untuk melakukan *noise filtering.*
3. Logitec Premium Headset, *microphone* ini mewakili jenis *microphone* dengan kualitas tinggi. Headset ini dibuat khusus untuk

melakukan *voice chat* dan *speech recognition* dengan komputer dan dilengkapi juga dengan *noise filtering.*

Berikut ini adalah hasil pengujian dari Aplikasi *Voice User Interface* dengan menggunakan berbagai jenis *microphone.*

Tabel 5.3 Hasil pengujian pengaruh kualitas *microphone* terhadap aplikasi *Voice User Interface*

Microphone	Prosentase ketepatan pengenalan suara	
	Mode Command	Mode Dictation
SONY Dynamic Microphone VJ-F22/C	96 %	90,0 %
SPEED SP-501 Headset	100 %	92,5 %
Logitec Premium Headset	100 %	92,5 %

Keterangan :

- Pengujian dilakukan dalam ruangan tertutup rapat pada waktu malam hari
- Pengujian dilakukan setelah melakukan pelatihan pengenalan suara sebanyak tiga kali

Efektifitas dan Tanggapan Responden

Tujuan akhir dari aplikasi VUI ini adalah memberikan alternatif media komunikasi antara manusia sebagai user dengan komputer. Walaupun aplikasi ini belum sepenuhnya dapat menggantikan aplikasi GUI, tapi setidaknya aplikasi VUI ini dapat meningkatkan efektifitas yang pada akhirnya dapat diharapkan dapat meningkatkan produktifitas. Untuk mengetahui seberapa besar efektifitas yang didapat dengan menggunakan aplikasi VUI ini dilakukanlah survei dengan berbagai responden yang berbeda, para responden dipilih dari orang-orang yang memiliki kemampuan Bahasa Inggris dan pengetahuan tentang komputer yang berbeda-beda.

Setiap responden menjalani 5 tahap pengujian guna mengetahui seberapa besar pengaruh pelatihan pengenalan suara terhadap jalannya aplikasi *Voice User Interface* dan untuk mengetahui apakah responden mendapatkan keuntungan atau kemudahan dengan menggunakan aplikasi *Voice User Interface* ini. Kelima tahap pengujian tersebut adalah sebagai berikut :

1. Pengujian aplikasi *Voice User Interface* tanpa melakukan pelatihan pengenalan suara terlebih dahulu. Pengujian ini dilakukan dengan memberikan 25 perintah dalam *mode command* dan membaca 40 kata dalam *mode dictation.*
2. Pengujian aplikasi *Voice User Interface* setelah sekali melakukan pelatihan pengenalan suara. Pengujian ini dilakukan dengan memberikan 25 perintah dalam *mode command* dan membaca 40 kata dalam *mode dictation.*
3. Pengujian aplikasi *Voice User Interface* setelah dua kali melakukan pelatihan pengenalan suara. Pengujian ini dilakukan dengan memberikan 25 perintah dalam *mode command* dan membaca 40 kata dalam *mode dictation.*
4. Pengujian aplikasi *Voice User Interface* setelah tiga kali melakukan pelatihan pengenalan suara. Pengujian ini dilakukan dengan memberikan 25 perintah dalam *mode command* dan membaca 40 kata dalam *mode dictation.*
5. Pengujian untuk mengetahui tanggapan responden terhadap aplikasi *Voice User Interface* dan apakah responden merasa mendapatkan keuntungan atau kemudahan jika menggunakan aplikasi *Voice User Interface* ini.

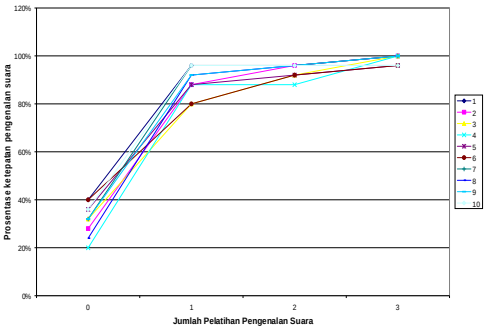
Berikut ini adalah hasil pengujian efektifitas dan tanggapan responden terhadap Aplikasi *Voice User Interface* yang dilakukan pada 10 responden :

Tabel 5.4 Hasil pengujian terhadap responden pada *mode command*

Responden ke	Jumlah Pelatihan Pengenalan Suara			
	0	1	2	3
1	40 %	96 %	96 %	100 %
2	28 %	88 %	96 %	100 %
3	32 %	80 %	92 %	100 %
4	20 %	88 %	88 %	100 %
5	36 %	88 %	92 %	96 %
6	40 %	80 %	92 %	96 %
7	32 %	96 %	96 %	100 %
8	24 %	92 %	96 %	96 %
9	32 %	92 %	96 %	100 %
10	36 %	96 %	96 %	96 %

Keterangan :

- Pengujian dilakukan dengan menggunakan Logitec Premium Headset



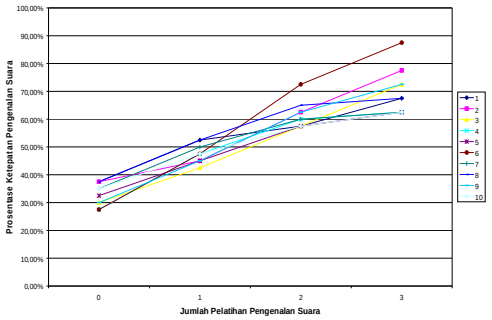
Gambar 5.2 Grafik hasil pengujian responden pada *mode command*

Tabel 5.5 Hasil pengujian terhadap responden pada *mode dictation*

Responden ke	Jumlah Pelatihan Pengenalan Suara			
	0	1	2	3
1	37,5 %	52,5 %	57,5 %	67,5 %
2	37,5 %	45,0 %	62,5 %	77,5 %
3	30,0 %	42,5 %	57,5 %	72,5 %
4	27,5 %	47,5 %	60,0 %	62,5 %
5	32,5 %	45,0 %	57,5 %	62,5 %
6	27,5 %	47,5 %	72,5 %	87,5 %
7	35,0 %	50,0 %	60,0 %	62,5 %
8	37,5 %	52,5 %	65,0 %	67,5 %
9	30,0 %	45,0 %	62,5 %	72,5 %
10	35,0 %	47,5 %	57,5 %	62,5 %

Keterangan :

- Pengujian dilakukan dengan menggunakan Logitec Premium Headset



Gambar 5.3 Grafik hasil pengujian responden pada *mode dictation*

VI. PENUTUP

Kesimpulan

1. Aplikasi *Voice User Interface* ini menjadi alternatif media komunikasi antara manusia sebagai *user* dan komputer.
2. Aplikasi *Voice User Interface* sangat terpengaruh kondisi lingkungan, kondisi lingkungan yang ideal untuk menjalankan aplikasi *Voice User Interface* adalah kondisi lingkungan yang tenang.
3. Berdasarkan hasil pengujian pengaruh kualitas *microphone* terhadap aplikasi *Voice User Interface*, disimpulkan hal-hal berikut :
   a. Penggunaan SONY Dynamic Microphone VJ-F22/C menghasilkan prosentase ketepatan pengenalan suara sebesar 96 % pada *mode command* dan 90,0 % pada *mode dictation.*
   b. Penggunaan SPEED SP-501 Headset menghasilkan prosentase ketepatan pengenalan suara sebesar 100 % pada *mode command* dan 92,5 % pada *mode dictation.*
   c. Penggunaan Logitec Premium Headset menghasilkan prosentase ketepatan pengenalan suara sebesar 100 % pada *mode command* dan 92,5 % pada *mode dictation.*
4. Semakin sering dilakukan pelatihan pengenalan suara maka prosentase ketepatan pengenalan suara oleh aplikasi *Voice User Interface* akan semakin mendekati 100%.
5. Aplikasi *Voice User Interface* yang berjalan pada *mode command* memiliki prosentase ketepatan pengenalan suara yang hampir mendekati 100 %, hal ini disebabkan karena kata-kata yang akan didengarkan oleh *speech recognizer engine* telah ditetapkan terlebih dahulu.
6. Aplikasi *Voice User Interface* yang berjalan pada *mode dictation* memiliki prosentase ketepatan pengenalan suara yang kurang baik, hal ini disebabkan karena *speech recognizer engine* menggunakan seluruh kata dalam database di dalam *grammar*-nya sehingga pengucapan yang salah sedikit saja akan menyebabkan kesalahan pengenalan.

Saran

1. Aplikasi *Voice User Interface* ini perlu dikembangkan sehingga mampu mengakses struktur menu dan struktur *child window* dari aplikasi yang memiliki struktur kompleks.
2. Aplikasi *Voice User Interface* ini akan lebih bermanfaat lagi bagi masyarakat Indonesia bila menggunakan *engine* dalam Bahasa Indonesia.

DAFTAR PUSTAKA

1. Booch,G., Jacobson,I., dan Rumbaugh,J., *"The Unified Modeling Language User Guide"*, Addison-Wesley Longman Inc., Massachusetts, USA, 1999.
2. Suhendar,A., Gunadi,H., *"Visual Modeling Menggunakan UML dan Rational Rose"*, Informatika, Bandung, 2002.
3. Nugroho, Widodo, *"Tip dan Trik Pemrograman Delphi"*, Elex Media Komputindo, Jakarta, 2002.
4. Pamitrapati, Dita, Siahaan, Krisdianto, *"Trik Pemrograman Delphi"*, Elex Media Komputindo, Jakarta, 2000.
5. -----, http://www.microsoft.com/speech/
6. -----, http://www.cs.vt.edu/~perez/VUI.pdf
7. -----, http://www.ling.lu.se/sounds/tutorial/SpeechAnalysisTutorial.htm
8. -----, http://www.computerworld.com/SpeechRecognition.htm
9. -----, http://www.therationaledge.com/content/nov_03/t_modelinguml_db.jsp
10. ------, http://www.therationaledge.com/content/jun_03/f_umlintro_db.jsp
11. ------, http://www.therationaledge.com/content/sep_03/f_umlbasics_db.jsp
12. Long, Brian, *"Speech Synthesis & Speech Recognition Using SAPI 5.1"*, http://bdn.borland.com/article/0,1410,29583,00.html.
13. Long, Brian, *"Adding Speech Synthesis and Speech Recognition capabilities into Delphi applications using The Microsoft Speech API (SAPI)"*, http://bdn.borland.com/article/0,1410,29580,00.html
14. Long, Brian, *"Using the Windows API in Delphi"*, http://www.blong.com/articles/UsingWinAPI.html
15. Long, Brian, *"Automation In Delphi 5"*, http://www.blong.com/articles/AutomationDelphi5.html
16. Sateli, Babak, *"…simulate the pressing of keyboard keys?"*, http://www.swissdelphicenter.ch/torry/printcode.php?id=220
17. Stutz, Thomas, *"…Click on a button of another application?"*, http://www.swissdelphicenter.ch/torry/printcode.php?id=727
18. Spence, Rick, *"Object Oriented Programming in Delphi A Guide for Beginners"*, http://www.webtechcorp.co.uk/course/OOPGuide.html
19. Microsoft Research Team, *"Natural Language Processing"*, http://research.microsoft.com/nlp/
20. Cummings, Chris, *"An Introduction to hook procedure"*, http://delphi.about.com/library/weekly/aa101000a.htm
21. Microsoft, *"Win32 Developer's Reference"*.
22. Microsoft, *"Microsoft Speech SDK 5.1 Help"*.
23. -----, http://msdn.microsoft.com/library/enus/winui/WinUI/WindowsUserInterface/Resources/Menus/MenuReference/

Budhi Supriyono adalah mahasiswa Teknik Elektro konsentrasi Komputer dan Informatika. Saat ini sedang menjalani proses untuk menyelesaikan studi Strata 1 pada Jurusan Teknik Elektro Fakultas Teknik Universitas Diponegoro Semarang.

Mengetahui/Menyetujui

Pembimbing I	Pembimbing II
Agung BP, ST MIT	**Adian FR, ST MT**
NIP. 132 137 932	NIP. 132 205 680