音声言語情報処理 15－4
（1997. 2. 7）

音声合成／認識APIと応用ソフト開発環境の動向

桃﨑 浩平, 原 義幸, 正井 康之, 松浦 博, 新田 恒雄
(株)東芝 マルチメディア技術研究所

パーソナルコンピュータの高速化とオーディオチップの標準搭載により, 音声合成・認識機能がソフトウェアだけで実現できるようになった. これにともなって, 音声を利用したアプリケーションソフトウェアが容易に開発できるような環境が整いつつある.

本文では, 標準APIの一つである米マイクロソフト社のSAPIを中心に紹介するとともに, OCXを用いる応用ソフトウェア開発について解説する.

Trends on Speech APIs and the Environment for Developing Application Software

Kohei Momosaki, Yoshiyuki Hara, Yasuyuki Masai, Hiroshi Matsu'ura, and Tsuneo Nitta
Multimedia Engineering Laboratory, Toshiba Corporation

Contemporary PCs provide sufficient computer power to accommodate Text-to-Speech (TTS) and Speech Recognition (SR) with no additional hardware. On the other hand, the environment for developing speech application software still remains in a big issue.

In this paper, we first explain the Microsoft Speech API, as one of the standard APIs for speech technologies, and then present some examples of application software using SAPI and speech OCX.

1. はじめに

最近のワークステーション（WS）やパーソナルコンピュータ（PC）は処理速度が飛躍的に向上し, またオーディオ入出力チップを標準搭載している. このため音声合成・認識技術はソフトウェアだけで実現できるようになった. そこでアプリケーションを簡単に構築することを目標に, 音声合成・認識機能を基本ソフトウェアに統合して, 複数のアプリケーションから利用[1]できるようにする仕組みが検討されている. 近年, 文音声合成本体（TTS: Text-To-Speech エンジン）あるいは音声認識本体（SR: Speech Recognition エンジン）の開発者（エンジンベンダ）とアプリケーション開発者が共に準拠できる標準のインタフェースとして, API (Application Programming Interface) が提供され始めている. これによって, アプリケーション開発者はエンジンの種類によらずプログラムが書け, またエンジンベンダはあらゆる応用システムに共通のエンジンを提供することができる.

本稿では API の紹介と TTS および SR への組み込み, ならびにアプリケーションの開発例を中心に説明する.

2. Microsoft 社 Speech API (SAPI)[2]

2.1 概 要

音声合成・認識を対象とした API は, これまで米国を中心に多くのものが実用に供されている. この中で Microsoft 社の SAPI[3]とエンジンベンダ中心の SRAPI[4] (Speech Recognition API) がよく知られている. ここでは前者の SAPI の概要を説明する. SAPI は Windows 95 および Windows NT オペレーティングシステム用に Microsoft 社が設計・提唱する, 文音声合成 (TTS) と音声認識 (SR) の API である.

SAPI には大きく分けて２つのレベルの API がある．１つは低水準 API で，エンジンに対する詳細な制御によって高度な処理が可能になる反面，API の利用方法はかなり複雑である．もう１つは高水準 API で，低水準 API をもとに簡単な制御のみを提供することによって実装が容易になる．高水準 API は単純な音声コマンドやテキスト音声合成に適しており，VB (Visual Basic) や VBA (Visual Basic for Applications) からもアクセスすることができる（2.3 参照）．

2.2 SAPI の仕組み

SAPI は従来の Windows API や拡張 API のような単純な関数セットとは異なり，COM (Component Object Model) [5]に沿ったオブジェクトの集合として定義されている．これによって実行ファイル（バイナリ）として存在するオブジェクトの管理が容易になり，例えば複数のベンダが提供するエンジンのバージョン管理などの問題が解決される[6]．COM に基づくことにより，ＯＳに変更を加えることなく，TTS・SR や関連する新しいサービスを追加することができる．

エンジン自体のオブジェクトや関連するいくつかのオブジェクトはエンジンベンダから，またエンジンに依存しないオブジェクトは Microsoft から提供される．

オブジェクトは参照されたときにプログラムの実体が呼び出され，メンバ関数が動的にリンクされる．関連するメンバ関数は「インタフェース」という形でグループ化され，インタフェースポインタを介して間接参照されるようになっている（図 1）．機能や仕様の異なる複数のインタフェースを１つのオブジェクトが持つこともできる．

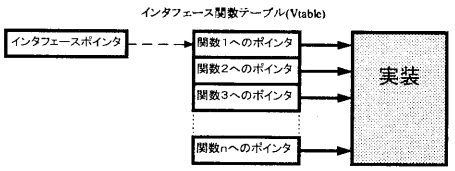

図１　バイナリインタフェース構造

オブジェクトやインタフェースは，仕様が異なる毎にユニークな識別子 GUID (Global Unique Identifier) が与えられる．GUID はツールやライブラリを使って自由に生成できるので，誰もが独立にサービス（機能）を定義して実装できる．また，誰かが定義したサービスの仕様に従って多くの人々がオブジェクトやクライアントを実装できることになる．

例えば，現在の SAPI でサポートされていない機能を新しくカスタムインタフェースとして追加すれば，エンジン固有の機能が使えるだけでなく，もしエンジン間で合意が取れれば日本語用の拡張とすることも可能である．

2.3 VB などのスクリプトからの利用

VB などのスクリプト言語でオブジェクトにアクセスする仕組みとして OLE オートメーションがある．OLE オートメーションオブジェクトは実行時にクラスの型情報を提供する機能を持っているのが特長である．

SAPI の高水準 API は，OLE オートメーションをサポートしている．設定や取得が可能な属性値「プロパティ」（例えば TTS の発話速度）と，メンバ関数に相当する「メソッド」（例えば TTS の読み上げ中止命令）を利用して，VB などでプログラムを作成できる．

2.4 アプリケーションからのエンジンの選択

SAPI は，言語等の異なる複数のエンジンをシステム内にインストールして利用することも想定している．また，表１に示すような「モード」を１つのエンジンが複数用意してサービスすることもできる．エンジンやモードはそれぞれ GUID で識別される．

低水準 API には，エンジンやモードを選択するための列挙（Enumeration）オブジェクトがあり，以下に述べる２種類のインタフェースのいずれかを利用して検索を行う．

表1　TTS と SR のモード

ＴＴＳ モード 情報 (TTSMOD EINFO)	メーカ名，製品（エンジン）名，モード名 言語・方言，話者の名前・性別・年齢 サポート内容（ＩＰＡ，制御タグ，声の高さ， 発話速度，音量，電話対応，Visual 通知） オプションインタフェースのサポート情報
ＳＲ モード 情報 (SRMODE INFO)	メーカ名，製品（エンジン）名，モード名 言語・方言，認識分類（連続，孤立単語等） 最大語彙数，グラマー種類 サポート内容（任意語彙，不特定話者・マイク，複数言語，電話対応，音韻・単語学習） オプションインタフェースのサポート情報

1つはモードの個々の情報に相対的ランク付けをして最適なエンジンを自動的に捜すものである（ITTSFind と ISRFind）．見つかったモード GUID によりエンジンのインタフェースを取得する．

もう1つは列挙により個々のエンジン(モード)の情報を取得した後，好みのモードを選択するものである（ITTSEnum と ISREnum）．

エンジンがそれぞれ自分のモードに関する上記のインタフェースを備えることにより，インストールされている全てのエンジンのモードが検索される仕組みになっている．

2.5　オーディオ入出力の仕組み

従来の Windows サウンド API に代わって COM ベースのオーディオ入出力（マルチメディアオーディオオブジェクト）が提供される．これにより，エンジンはそのままでオーディオ入出力まわりのカスタマイズが可能になる．例えば WAV ファイルとして音声を入出力したり，残響効果を出力音声に施すなどの機能を持つ COM オブジェクト(カスタムオーディオオブジェクト）を作成してマルチメディアオーディオオブジェクトの代わりに利用することもできる．

2.6　PC電話アプリケーションとの連携

ＰＣと電話の統合のための標準インタフェースとして TAPI (Telephony API) [7]があり，モデム毎の専用 API を使わずに電話回線の制御機能が利用できる．

アナログ電話回線用のモデムの中には，データやＦＡＸの送受信の他に通常の電話音声を入出力できるものがある．また，ＤＳＰを用いたＰＣ電話ボードなどもつくられている．このようなボイス機能に対応した汎用ドライバ(UnimodemV) が新たに Windows 95 に追加[8]されたことにより，TAPI を利用した留守番電話やボイスメールなどのアプリケーション開発[9]が可能となった．ダイアル発信や応答により回線接続が確立したあとは Windows のサウンド API 等で電話回線の音声入出力ができる．ここで SAPI による TTS や SR を利用すれば，かなり複雑な電話音声応答サービスも可能で，英語版に関してはすでにこのようなＰＣ電話アプリケーションがモデム等にバンドルされて市販されている．

3. TTS の組み込み例

3.1　全体の処理の流れ

単純な TTS 機能を簡単に利用するための高水準 API として，音声テキスト（VoiceText）オブジェクトが提供されている．内部では低水準 API によって各ベンダの TTS エンジンを駆動して動作する（図2）．

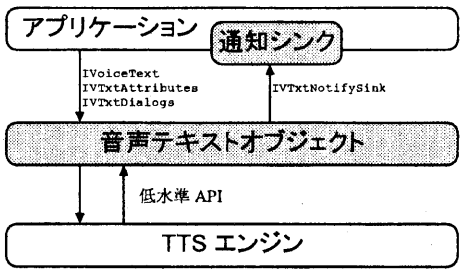

図2　音声テキストオブジェクトの構造

音声テキストオブジェクトの基本的な利用方法は次のとおり．

(1) 音声テキストオブジェクトを作成．

(2) アプリケーションへ音声出力の経過などを通知するための通知シンク（3.6 参照）として COM オブジェクトを作成．

(3) 出力先デバイス，アプリケーション名，通知

シンクを登録.
(4) テキストを喋らせる.
(5) 通知シンクを介して，しゃべり終わったときなどにアプリケーションに通知される.

なお，通知シンクはオプションで，(2)および(5)は省略できる.

3.2 高水準 API を用いたアプリケーション例

例えば，高水準 API を用いて次のような機能をもつアラーム時計ができる.
(1) 与えるテキストによって時刻の読み上げ方をいろいろ変えられる.
(2) しゃべっている間だけ絵を表示する. `IVTxtNotifySink::SpeakingDone` を使用して実現する.
(3) 時報はゆっくり，アラームは早口にする. `IVTxtAttributes::SpeedSet` を使用して実現する.
(4) エンジン設定のダイアログボックスを開いて，声の高さなどを変えたりできる. `IVTxtDialogs::GeneralDlg` を使用して実現する.

3.3 OLE オートメーションを用いた例

PC では近年 VB 等のビジュアル言語を用いてアプリケーションを作成することが多くなっている. また，同様の文法を備えたスクリプト言語 VBA による機能拡張をサポートするアプリケーションが提供されている. これらの上で TTS プログラムを作成する場合には，OLE オートメーションにより高水準 API を利用できる.

図 3 は VB による OLE オートメーションプログラムの例で，ウィンドウ上のボタンをクリックすると "Hello, world!" と喋る. 同時に，画面に，喋っている間は "Speaking"，喋り終わると "Done" と表示される.

図 3 VB による OLE オートメーションの例

```
' フォーム上のボタンCommand1 クリック時の処理
Private Sub Command1_Click()
  Dim Vtxt As Object
  Set Vtxt = CreateObject("Speech.VoiceText")' VoiceText オブジェクト
  Call Vtxt.Register("","VoiceText Test") ' サイト(出力先 アプリ名) 登録
  Label1.Caption = "Speaking"              ' フォーム上にテキストを表示
  Call Vtxt.Speak("Hello, world!", &H2016)
  Do While Vtxt.IsSpeaking ' 発話終了したかどうかくり返し調べる
  Loop                     ' 実際にはCPU時間を浪費しない処置が望ましい
  Label1.Caption = "Done"
  Set Vtxt = Nothing       ' オブジェクトを解放
End Sub
```

3.4 低水準 API の利用

低水準 API では，プログラムで声の高さなどを変えたり，エンジンの独自機能を使ったりできる. テキスト中に制御タグ (3.5 参照) を埋め込んで，詳細な制御をすることもできる. また，2.4 で述べたように，エンジン (モード) を捜す機能は低水準 API の一部として提供されている. さらに，2.5 で述べたカスタムオーディオオブジェクトをエンジンに渡して，TTS 出力のカスタマイズができる.

3.5 制御タグ(TTS Control Tag)

制御タグはバックスラッシュ文字 (\) で囲まれた形式の文字列である.

低水準 API では，テキスト中に制御タグを埋め込んでおくことで TTS のさまざまな属性を変更しながら読み上げさせることができる (図 4). また，\Mrk=1\ のようなブックマークのタグを埋め込み，その位置でアプリケーションへ通知を送らせることもできる. エンジン独自機能もタグを定義して利用できるようになっている.

```
\Vce=Gender="Female"\こちらは新幹線予約センター です。\Mrk=1\
```

図 4 制御タグを用いたテキストの例

なお，高水準 API では，タグの埋め込みはできないが，IVTxtAttributes インタフェースでサポートされていない TTS 設定を Speak の引数としてタグ形式で指定できる．

3.6 通知シンク(NotifySink)

TTS は多くの場合非同期で動作し，音声出力が完全に終わる前にエンジンから制御が戻る．アプリケーションは，エンジン側から呼び出される通知シンクを用意することにより，音声の出力開始・終了など，必要なタイミングで情報を得ることができる．音韻や口の形などをリアルタイムに通知する機能 (Visual)も定義されているので，エンジンがサポートしていれば，音声の出力に合わせて口が動くアニメーションなどが実現できると考えられる．

4. SR の組み込み例

4.1 全体の処理の流れ

音声によるコマンドの入力などの単純な音声認識機能を利用するための高水準 API として，音声コマンド (VoiceCommand) オブジェクトが提供されている．内部では低水準 API によって各ベンダの SR エンジンを駆動して動作する（図 5）．

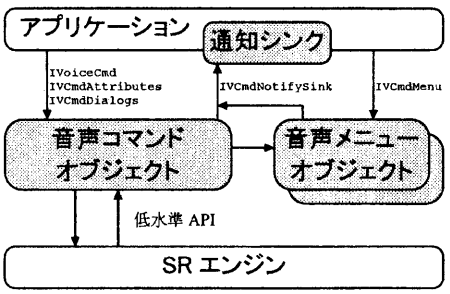

図 5　音声コマンドオブジェクトの構造

音声コマンドオブジェクトの基本的な利用方法は次のとおり．

(1) 音声コマンドオブジェクトを作成する．
(2) 音声コマンドオブジェクトからアプリケーションへ認識結果などを通知するための通知シンクとして COM オブジェクトを作成する．
(3) 認識語彙を表現する音声メニュー（VoiceMenu）オブジェクトを作成する．
(4) 認識させたい音声メニューオブジェクトをアクティブにすると，そのメニューに含まれる語彙が認識対象となる．
(5) 発声すると通知シンクを介して認識結果がアプリケーションへ通知される．認識結果を通知するために使用する関数 CommandRecognize は，アプリケーションが提供するコールバック関数である．

4.2 高水準 API を用いたアプリケーション例

高水準 API を用いて次のような機能を持つ「おしゃべり時計」を作ることができる．

(1) 音声認識の起動コマンド「Wake Up」を認識すると，「おしゃべり時計」が提供する音声コマンドの「いま何時」と「きょうは何日」を認識できるようになる．
IVCmdAttributes::AwakeStateSet を使用して実現する．
(2) ウィンドウのフォーカスの有無に関係なく，音声コマンドの「いま何時」と「きょうは何日」を認識する．但し「時刻設定」はおしゃべり時計にフォーカスが当たっているときだけ認識する．
IVCmdMenu::Activate を使用して実現する．
(3) アラームの時刻設定のように誤設定を避ける必要のある場合は，認識結果をユーザに確認してから設定する．
IVCmdDialogs::CommandVerifyDlg を使用して実現する．

4.3 低水準 API の利用

高水準 API を使用すればほとんどのアプリケーションを構築できるが，ディクテーションなど複雑な認識を実装したい場合には，低水準 API を使用しなければならない．低水準 API では認識結果（SRResults）オブジェクトが提供され，複数の認識結果や話者識別結果を取得することができる．ただし，ほとんどの機能はオプションで，エンジンに依存している．

5. OCXによるアプリケーションの開発

5.1 OCXの概要

OCXは，VBなどで用いられるカスタムコントロールの一種で，2.3で述べたOLEオートメーションを拡張したものである．OCXを組み込むことにより，各々のもつプロパティやメソッド，そして通知シンクに相当する「イベント」の機能が簡単に利用できるため，拡張ライブラリ的にも用いられる．OLEオートメーションとの違いは，VB上でデザインする際プロパティを直接設定できる，画面表示される部品にできる，等である．

OCX自体はSAPIと直接関係ないが，アプリケーション開発上有効と思われるので簡単に紹介する[10]．

5.2 アプリケーションの開発例

図6は，「いま何時？」と話しかけると「10時10分です」等と現在時刻をしゃべる音声応答時計アプリケーションの例である．

```
' 起動時の処理
Private Sub Form_Load()
   SR.Initialize hwnd          ' SRの初期化
   SR.AddList "今何時,いまなんじ" ' 認識語彙（読み）の設定
End Sub
—
' 認識時の処理
Private Sub SR_CommandRecognize(Byval Phrase As String)
  If Phrase = "今何時" Then
    TimeNow = Time             ' 現在時刻の取得
    TTS.Text = Hour(TimeNow)&"時" & Minute(TimeNow)&"分です。"
                               ' 読み上げテキストは「○時○分です。」
    TTS.Speak                  ' 変換したテキストの読み上げ
  End If
End Sub
```

図6　VBによるOCXプログラムの例

この例は以下に示すようにTTSとSRのOCXを用いてVBで簡単に作成できる．

(1) TTSとSRのOCXをVB上のプロジェクトに組み込み，画面（フォーム）の上に配置する．
(2) プロパティで変化しないものを予め設定する．
(3) 起動時にSRの認識語彙を設定し，認識開始するメソッドを呼ぶ処理を追加する．
(4) 現在時刻を取得して読み上げのテキストを作成し，TTSの読み上げのメソッドを呼ぶ処理をSRの認識イベントに追加する．

参考文献

[1] 松浦博ほか，“東芝における最近の音声合成・認識の応用，”音声言語情報処理研究報告 96-SLP-12-8 (1996)

[2] Mike Rozak, “Microsoft Speech APIでコンピュータに話しかけ，応答を得る，” MSJ No.42 (1996)

[3] *Speech API Developer's Guide Version 2.0* (Microsoft , 1995)

[4] *Speech Recognition Application Program Interface Specification* (SRAPI Committee, 1995)

[5] Kraig Brockschmidt, 長尾高弘訳, *Inside OLE2* (アスキー, 1994)

[6] Kraig Brockschmidt, “コンポーネントソフトウェアデザインに対するOLEとCOMの解答，” MSJ No.44 (1996)

[7] Charles Mirho and Andrew Raffman, “Windows Telephony APIがPCを結び付ける，” MSJ No.30 (1994)

[8] Charles Mirho, “Windows 95ボイスモデムエクステンション，” MSJ No.45 (1996)

[9] Michael C. Amundsen, *MAPI, SAPI, and TAPI Developer's Guide* (Sams Publishing, Indianapolis, 1996)

[10] 小野木智宏ほか，“音声認識／合成ＯＣＸの開発，”電子情報通信学会全国大会 D-456 (1996.9)

MSJ:マイクロソフトシステムジャーナル日本版 (アスキー)