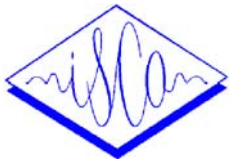

6th European Conference on
Speech Communication and Technology
(EUROSPEECH'99)
Budapest, Hungary, September 5-9, 1999

VOCAPI - SMALL STANDARD API FOR COMMAND & CONTROL

Eike Gegenmantel
Philips Speech Processing
Weisshausstrasse 2, 52066 Aachen, Germany
gegenman@pfa.research.philips.com
http://www.speech.philips.com

ABSTRACT

Some speech APIs are becoming standards to access speech functionality. They aim to be highly flexible to avoid restrictions. Consequently, corresponding interface specifications are extensive and implementations have high demands on the performance of processing platforms. Resulting prices are unacceptable for consumer devices like phones, car navigation, audio, TV, etc. Here, the state of the art requires low resource consumption while the range of functionality can be focused on command and control dialogs. A consortium of major companies has come together to develop and standardize a small and handy speech API, the so-called Voice Control API (VOCAPI). Being a German draft standard [1] now, VOCAPI will be proposed as European standard in the near future.

Keywords: speech API, standardization, command and control, consumer devices, real-time platforms

1. INTRODUCTION

Numerous consumer devices can take advantage of speech control. Additional benefit pushes the request for speech control in some applications, for instance a gain of safety in the car environment. Where now the driver has to touch buttons, devices like phone, audio and navigation system will be operated without the use of hands and eyes. A speech recognizer transforms audio input into digital statements. These are accepted and processed by an application program (refer to Figure 1) which controls the speech recognizer. The interface between the self-contained speech recognizer and the controlling application is called „Voice Control Application Programming Interface", short VOCAPI.

1.1 Motivation

Price pressure influences the development of consumer devices. This is the reason why in this environment resources like memory and processing power are always tight, operating systems - if available at all - often have restrictions. Real time conditions are common. VOCAPI distinguishes itself by meeting these demands and being independent of any operating system. The key is that VOCAPI focuses on command and control, while other APIs concentrate on other domains or cover a wide range of speech applications, including dictation with large vocabularies and language grammars. Well-known APIs like Java Speech API (JSAPI) [2] and Microsoft Speech API (MS SAPI) [3] even support the selection of different speech engines for different purposes. This high degree of flexibility is expensive in the aspect of resource consumption.

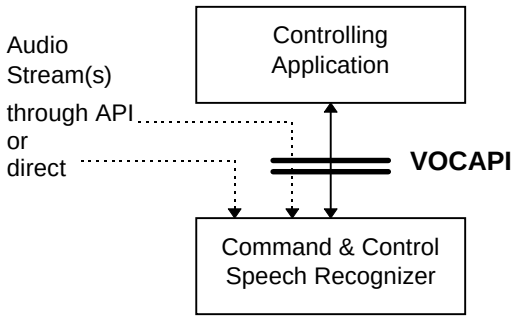

Figure 1: VOCAPI = Voice Control API

The standardization of VOCAPI gives advantages to all involved parties. Manufacturers of applications and devices become independent of certain speech recognizer implementations. Providers of speech technology see a chance to speed up the introduction of speech control in automotive and consumer devices.

1.2 History

The definition of VOCAPI has its origin in a simple speech API of a single-word speech recognizer. While its simplicity has been convincing, requirements of continuous speech recognition had to be considered, leading to the first specification of VOCAPI developed by the MoTiV research initiative [4]. Following the wish of the MoTiV partners for standardization of VOCAPI, the speci-

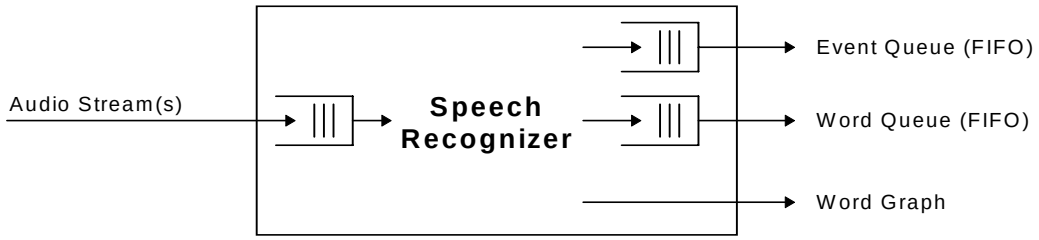

Figure 2. Data flow through speech recognizer

fication has been refined by the group „NI-Erg/UA6“ of the German Institute for Standardization (DIN) and is now available as draft standard [1]. The companies involved in standardization are Bosch, Daimler-Benz, IBM, Mannesmann/VDO, Opel, Philips, Siemens, Sony, and Volkswagen. They intend to involve further partners and to propose VOCAPI as a European standard (CEN) in the near future.

2. VOCAPI PRINCIPLES

VOCAPI is independent of the used platform, programming language and interface mechanism (e.g. functional or message based interface). The VOCAPI specification considers the mapping to different environments. If required, appropriate mapping rules may be appended to the core specification to reduce the room for interpretation.

2.1 Data Flow Concept

The speech recognizer is seen as a device that transforms audio input into words. Figure 2 shows the principal data flow: Audio samples are fed into the input queue (ring buffer) of the speech recognizer. When the input is processed, different events are generated like “begin of speech detected”, “word was recognized”, “end of speech detected”, etc. For each event a notification is written into a queue where it is available for the application program.

Each time the recognizer computes a word from the audio input, this word is written into a word queue. From different recognition alternatives only the acoustically best choice is taken. An application program can read the queue word by word to process the recognition results.

Alternatively to the word queue, an application can get full information about a recognition result by a word graph. The word graph is only available after processing a complete utterance. It contains several recognition possibilities like alternative words, word sequences or even a network of words and sub-phrases to each of which the acoustic probability may be indicated. With the word graph an application has all possibilities to interpret the recognition result by use of further knowledge such as language models and context information.

2.2 The Recognition State Machine

Depending on the current mode of operation, the recognizer and with it the API of the recognizer can take different states. The API state has to be considered for several of the API functions, since some of them influence the state, while others will only work properly in certain states. The transitions between API states are triggered by events. All events generated in the recognizer are written into an event queue. This queue can be read out by the application to follow the complete event history.

The possible states and transitions are shown in Figure 3. The given diagram can be modified: The RECORDING state is omitted for recognizers that do not support the training of user defined words. A REPLAYING state and corresponding events are added for recognizers that support the replay of announcements.

3. THE API FUNCTIONS

This section gives an overview on the complete set of VOCAPI functions. They are grouped into packages. Some of them are mandatory for the operation of speech recognizers while others may be added optionally, depending on the recognizer’s capabilities and implementation requirements.

3.2 General Functions

The general functions comprise basic operations like checking the API version, opening and closing the API, handling of recognition parameters and observing of API states and recognition events.

```
Init( callBackFunction )
Exit()
GetAPIVersion( &versionString, &compatibilityStamp )
SetParameter( parameter, value )
```

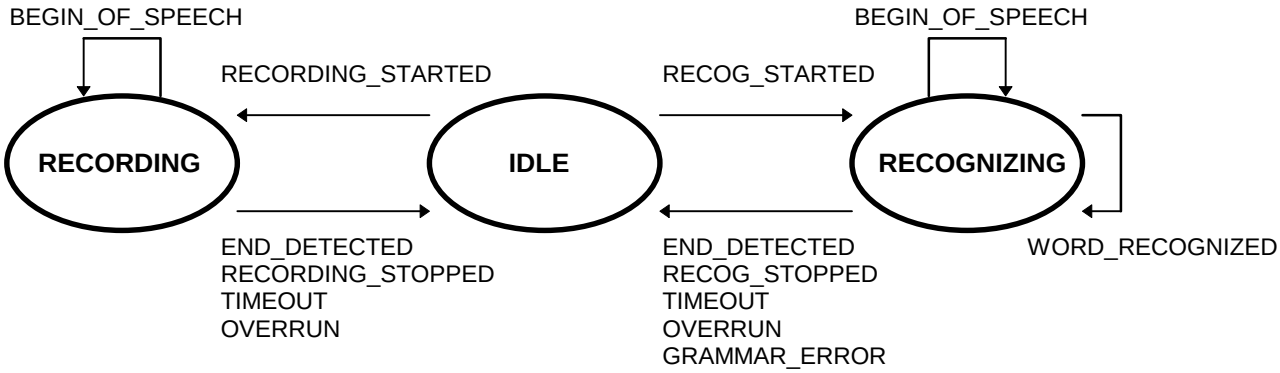

Figure 3. The VOCAPI state diagram

```
GetParameter( parameter, &value ) [1]
SetParamBlock( paramBlock )
GetParamBlock( &paramBlock, &length )
GetState( &state, &attention )
GetEvent( &event )
ClearEventQueue()
```

3.2 Vocabulary Functions

A vocabulary is defined as a set of words including their corresponding acoustic descriptions. The definition of a word is a recognition unit which may correspond to a usual word, but also to phonemes, pause, or garbage models. The recognizer may keep several vocabularies at a time. They may be fixed in the recognizer's implementation or dynamically loaded and removed through the API. The internal format of a vocabulary description is defined by the manufacturer.

```
StructGetVocs( &vocIdsSI, &vocsIdsSD )
VocGetVersion( vocId, &vocName, &compatStamp )
VocGetAllWords( vocId, &wordList )
VocWordGetParameter( word, parameter, &value )
VocWordSetParameter( word, parameter, value )
VocSet( vocabulary, &vocId )
VocGet( vocId, &vocabulary, &length )
VocDelete( vocId )
VocSetActiveWords( wordList )
VocAdapt( vocId, wordList, bufferId )
```

3.4 Recognition Functions

Functions of this set control the recognition process and retrieve recognition results. The recognizer begins to process audio input with the call of the start function. It will stop by itself or can be forced to stop. If the recognizer works in continuous operation, the controlling program is informed on intermediate results by events (with GetEvent(), general package). Single words can be retrieved one by one while recognition is in progress. The complete recognition result can be obtained as a word graph after the recognition process has stopped.

```
RecogStart()
RecogStop()
RecogGetWord( &word )
RecogGetGraph( &edges )
```

3.5 Processing Functions (optional)

The processing package offers interface functions to provide the recognizer with audio input and processing time. The package is not needed if the speech recognizer is implemented as a self-contained unit with own processing capabilities and audio device. If processor and/or audio device have to be shared, VOCAPI offers the necessary interface.

The processing functions can either be called periodically by the application program or used as entry points for interrupt handlers. The later scenario allows to operate the recognition (and training) processes as background tasks without the necessity of sophisticated operating system functions.

```
HRT( samples, channelId )
SRT()
ClearSampleBuffers()
```

3.5 Language Model Functions (optional)

Different recognizers have different language model capabilities. Simple recognizers may support a simple zerogram only, i.e. they can select a set of equally probable words (by VocSetActiveWords(), vocabulary package). More sophisticated recognizers support more detailed models. VOCAPI offers an optional package of functions which allows to handle unigrams, a finite state grammar (FSG), and a combination of these two,

[1] The '&' sign indicates parameters by reference to allow the return of a result.

where for each grammar state the words may be weighted by a related unigram.

```
FsgGetFreeState( &stateId )
FsgNewState( stateId, wordList, targetStateList,
                frequencyList )
FsgDeleteState( stateId )
FsgAddWord( stateId, word, targetStateId, frequency )
FsgDeleteWord( stateId, word )
FsgCheck( &stateId, &word, &targetStateId )
```

3.5 User Word Functions (optional)

Some recognizers enable the user to train new words at runtime. These "user words" are stored in speaker-dependent vocabularies. VOCAPI offers an optional package with functions to support the training of user words. It extends the vocabulary package to allow the maintenance of speaker-dependent vocabularies.

```
VocNew( vocName, &vocId )
TrainInit( word )
TrainRecordStart()
TrainRecordStop( accept )
Train( accept )
VocWordDelete( word )
```

3.5 Announcement Functions (optional)

Speech engines that support the replay of announcements can use this optional VOCAPI package. Announcements are stored in audio buffers which can be filled with pre-produced announcements or recordings of recognition or training utterances. For replay, a sequence of audio buffers can be specified.

```
AudioGetFreeBuffer( &bufferId )
AudioSet( bufferId, audio )
AudioGet( bufferId, &audio, &length )
AudioPlay( bufferIdList )
AudioStop()
AudioDelete( bufferId )
```

4. RELATION TO OTHER SPEECH API'S

General purpose speech APIs like JSAPI [3] and MS SAPI [2] are becoming de facto standards. Compared to them, VOCAPI is not „yet another" speech API and does not compete these interfaces. VOCAPI is small, simple and specialized in command & control. Extended features like coordination and dispatch of speech input for multiple simultaneous applications as well as dialog control are seen as functions to be placed at higher interface layers and kept out of VOCAPI.

A VOCAPI speech recognizer may be used by the implementation of a general purpose speech API. If a general purpose API offers a choice for command & control speech recognition there is now the choice to realize a „VOCAPI socket" instead of using a fixed command & control speech engine. Different implementations of VOCAPI speech engines could be plugged in.

5. AN IMPLEMENTATION OF VOCAPI

The Philips Voice Control speech recognizer „VoCon" realizes the VOCAPI interface as its native API. It shows that a code size of not more than 150 kBytes suffices for the API including the speech recognizer - depending on platform and optimization, the size can be reduced again. The implementation proves the usability for different platforms and environments. VoCon offers VOCAPI as a C interface as specified in an informative appendix to the VOCAPI draft standard which illustrates the mapping of VOCAPI for the C programming language. For VoCon, also other mappings of VOCAPI are planned, as for instance an ActiveX interface and a message-based realization for serial inter-IC-communication.

6. CONCLUSION

On its way to a European standard, the command and control speech interface VOCAPI has been developed by a consortium of major companies and is a German draft standard now. VOCAPI is designed for implementations with restricted and limited resources, as such for automotive and consumer devices. General purpose speech APIs can also make use of VOCAPI as a standard way of accessing command and control speech recognition engines. Philips provides an implementation of VOCAPI in C. Further implementations by other companies and for other programming environments will follow.

7. REFERENCES

[1] Deutsches Institut für Normung (1999), Programmierschnittstelle für Sprachsteuerung (VOCAPI), draft standard, DIN 33880.

[2] Microsoft Corporation (1999), Microsoft Speech API, Version 4.0.

[3] Sun Microsystems (1998), Java Speech API Programmer's Guide, Version 1.0.

[4] http://www.motiv.de