

Project P923-PF

Multilingual WEB sites: Best practice, guidelines and architectures

Deliverable 1

Guidelines for building multilingual Web Sites

Volume 4 of 5: Annex C

Applications of speech technology

Suggested readers:

This document is primarily aimed at anyone who is involved in the process of designing, building or managing WEB sites. It is of immediate relevance to those involved with multilingual WEB sites, but it nevertheless, provides information which will allow monolingual WEB site designers to design sites that are economically upgraded to multilingual sites.

EDIN 0010-0923

Project P923

For full publication

September 2000

EURESCOM PARTICIPANTS in Project P923-PF are:

- Koninklijke KPN N.V.
- France Télécom
- British Telecommunications plc
- Telecom Italia S.p.A.
- Portugal Telecom S.A.

This document contains material which is the copyright of certain EURESCOM PARTICIPANTS, and may not be reproduced or copied without permission.

All PARTICIPANTS have agreed to full publication of this document.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the PARTICIPANTS nor EURESCOM warrant that the information contained in the report is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using this information.

This document has been approved by EURESCOM Board of Governors for distribution to all EURESCOM Shareholders.

Executive Summary

The World Wide Web offers information available for anyone who has access to the Internet. So far access to this information happens via traditional computers. Voice may also be offered as an adjunct to conventional desktop browsers with high resolution graphical displays, providing an alternative browsing method opposing to the use of keyboard or screen.

Voice Browsers exploit voice input and output using a combination of keyboard and speech recognition for input, and pre-recorded file or speech synthesis for output. In this way the users can have access to the Web anytime and anywhere (at home, on the move or at work). By using only a cellular phone or POTS, they can choose to respond or reject by a key-press or a spoken command.

The aim of this document is to provide an inventory of the tools that allow voice access to Web pages, a view on markup programming languages used to develop Web page browsing by voice, and a description of the scientific community's activities for developing Voice Browsers.

List of Authors

Carolina Di Cristo, Cselc SpA

Contents

Executive Summary	1
List of Authors	2
Abbreviations	4
1 Applications of speech technology.....	6
1.1 Why voice browser and speech markup.....	7
2 Tools.....	8
2.1 Conversa Web	8
2.2 FreeSpeech Web.....	9
2.3 Web on Call.....	10
2.4 AudioWeb	11
2.5 speechWeb	12
2.6 pwTelephone.....	12
2.7 WebGALAXY	13
2.8 Speech Aware Multimedia.....	14
2.9 SpecHTML.....	15
3 Application development environments.....	17
3.1 VoxML.....	17
3.1.1 VoxML development platform.....	17
3.1.2 VoxML Software Development Kit.....	18
3.2 Mobile Application development Kit.....	20
3.3 VoiceXML	20
3.3.1 VoiceML architectural model	20
3.3.2 VoiceXML markup language.....	21
3.4 SpeechML.....	23
3.4.1 SpeechML architectural model	23
3.4.2 SpeechML markup language.....	24
3.4.3 The conversational browser.....	25
3.5 Java Speech API.....	26
3.5.1 Java Speech Grammar Format	27
3.5.2 Java Speech Markup Language.....	28
4 Research	30
4.1 W3C Voice Browsers activity.....	30
5 V-commerce	31
5.1 V-commerce development environment	31
6 Portals.....	33
6.1 Oracle Portal-to-Go.....	33
6.2 TelePortal	34

Abbreviations

ADML	AuDio page Markup Language
API	Application Programming Interface
ASR	Automatic Speech Recognition
CAD	Computer Aided Design
CASSI	Conversa Advanced Symbolic Speech Interpreter
CSS	Cascade Style Sheet
DARPA	Defense Advanced Research Projects Agency
DSP	Digital Signal Processors
DTMF	Dual Tone Multi-Frequency
ECMA	European Computer Manufacturers Association
GUI	Graphical User Interface
HDML	Hypertext Dynamic Markup Language
HTML	Hypertext Markup Language
HTTP	HyperText Transfer Protocol
IVR	Interactive Voice Response
JSAPI	Java Speech API
JSGF	Java Speech Grammar Format
JSML	Java Speech Markup Language
LAN	Local Area Network
MIT	Massachusetts Institute of Technologies
MMX	Multi-Media eXtensions
Mobile ADK	Mobile Application Developers Kit
MSAA	Microsoft Active Accessibility
PC	Personal Computer
PDA	Personal Digital Assistant
SAM	Speech Aware Multimedia
SGML	Standardised Generalised Markup Language
SN	Service Node
SQL	Standard Query Language
TTS	Text-to-speech
URI	Universal Resource Identifier
URL	Universal Resource Locator
USB	Universal Serial Bus
VoiceXML	Voice eXtensible Markup Language

W3C	World Wide Web Consortium
WAP	Wireless Application Protocol
WinCE	Windows CE
WMLS	Wireless Markup Language Script
WNT 4.0	Windows NT 4.0
WWW	World Wide Web
XML	Extensible Markup Language
Y2K	Year 2000

1 Applications of speech technology

Speech technology is becoming increasingly important in both personal and enterprise computing as it used to improve existing user interfaces and to support new means of human interaction with computers. Speech technology can allow hands-free use of computers and support access to many computing facilities away from the desk and over the telephone. Speech recognition and speech synthesis can improve accessibility for disabled users and reduce the risk of repetitive strain injury and other problems caused by current interfaces.

Here there is a brief description about the different areas of speech applications:

- **Telephony Systems**

Speech technology is already being used by many enterprises to handle customer calls to provide access to information and resources. Speech recognition is being used to provide a more natural and efficient interface than touch-tone systems for facilities such as voice mail, menu selections and entering numbers. Systems are available for telephone access to email calendars and other computing facilities that have previously been available only on the desktop or with special equipment. Such systems allow convenient computer access by telephone in hotels, airports and aeroplanes. Prototype telephone systems are providing a conversational style of interacting with these systems.

- **Desktop**

Speech technology can augment traditional graphical user interfaces. It can be used to provide audible prompts with spoken "Yes/No/OK" responses that do not distract the user's focus. Complex, structured commands can provide rapid access to features that are traditionally hidden in sub-menus. Drawing and CAD applications can be enhanced by using speech commands in combination with mouse and keyboard actions to improve the speed at which users can manipulate objects. Speech dictation systems are widely available and are already popular with users who have restricted typing ability.

- **Hands Free Computer Interfaces**

In many situations where keyboard input is impractical and visual displays are restricted or unavailable, speech provides the only way of interacting with a computer. For example, surgeons and other medical staff can use speech dictation to enter reports when their hands are busy and where touching a keyboard represents a hygiene risk. In vehicle and airline maintenance, warehousing and many other hands-busy tasks, speech interfaces can provide practical data input and output and can enable computer-based training.

- **Small Computing Devices**

Speech technology is being adopted in a wide range of small-scale and embedded computing devices to enhance their usability by recognising simple commands and producing synthesised speech. Such devices include Personal Digital Assistants (PDA), telephone handsets, toys and consumer product controllers. In some devices

the speech processing is implemented by dedicated hardware while in others Digital Signal Processors (DSP) capabilities are used. The Java Platform is being widely adopted in these same computing devices because of its compactness, robustness and portability - necessary capabilities to support access to constrained speech recognition and speech synthesis in this class of devices.

- Internet

Speech recognition can be used to control browsers, fill out forms, control Java applets and enhance the WWW/Internet experience in many other ways. Speech synthesis can be used to bring web pages alive, inform users of the progress of applets, and dramatically improve browsing time by reducing the amount of audio sent across the Internet.

1.1 Why voice browser and speech markup

Each of the preceding applications could be built without a speech markup language and a voice browser, however they provide several important capabilities:

1. Speech markup provides a higher-level tool that makes building conversational applications easier, in the same way that HTML allows visual applications to be built more easily. A speech markup language also reduces the amount of expertise in speech required.
2. A speech markup language can be designed to directly support networked (or web-based) applications - meaning applications where a user at one location is accessing information or an application provided by a server at another geographically or organisationally distant location. This delivers the connectivity and commerce of the web through the speech modality.
3. The use of a single browser to interpret streams of markup language originating from multiple locations provides the user with a seamless conversational experience across independent applications.

2 Tools

This section describes the features and the limits of the most representative commercial tools, which allow browsing the Web using voice. These came out not only from a reading of the documentation, but from a direct experience using the tools too.

2.1 Conversa Web

Conversa Web is a voice-enabled Web browser developed in 1998 by Conversa Corporation¹. Internet users activate Web Browsing and navigation by voice.

Conversa Web allows to use any combination of mouse, keyboard or voice commands at any time; if the user prefers to have only interaction with mouse or keyboard, he can say "Go to Sleep" or "Stop Listening" to instruct the browser to ignore all voice commands. Most Web pages will load slightly faster than if standard and voice commands are both active at the same time.

There is a set of commands for basic functions, such as "Go Back," "Scroll Down" and "Refresh page." By speaking the phrase "What can I say" the user obtains the repertoire of available commands. He/she can also scroll up and down at the desired speed, zoom in and out, print a page, move back and forth between pages, with the use of simple phrases. A new search starts by saying "Search the Web". Other commands such as "Show Favourites", "Show History List", "Link me to", "Add to Links toolbar", "Add to Favourites" are quite helpful to navigate the Web via voice. Elements of Web pages such as checkboxes, radio-buttons and drop-down menus can be voice activated; for entering URL addresses and text the Military Alphabet is used. A voice activated history list and a URL AutoComplete feature allow for fast retrieval of previously entered addresses. The SayConsTM voiceable icons allow to read toolbar commands, graphics, and other Web page elements e.g. Refresh Page is shown as a voiceable icon in the SayCons toolbar which appears in the page. A special numbered SayCon is inserted in the upper right corner of the image, so to activate the image link, you can simply say "number" followed by the actual number, avoiding the need of difficult-to-pronounce voice links. In many instances, the user needs only to say the first few words of a link. When the voice browser is certain which link the user is speaking, it will take him there immediately-without having to finish the sentence. There are some links that don't work well with voice. Some Java, Visual Basic applets, and server-side scripts that contain embedded links will need to be selected via mouse instead of voice. The user knows these instantly because they won't have SayCons next to them. Most all other links, whether text or image format, will be voiceable.

The toolbar, address bar and status bar are all configurable. The user may choose to view them in their entirety, view only portions of them, or not view them at all. For example, the toolbar can be configured to display only the SayCons, only the text, both the SayCons and the text, or none of the above. Becoming more familiar with Conversa Web, the user may choose not to display these items, thus vacating additional screen area for your Web pages. Because this tool is fully voice-operable, even though he can't see the SayCons, he can still say them at any time.

A Text-To-Speech (TTS) is available to read the screen and to listen to the objects on the screen. An on-line voice help includes commands to set the preferences and the

¹ www.conversa.com

options for audio controls settings. Conversa Web uses a proprietary speech recognition technology called Conversa Advanced Symbolic Speech Interpreter (CASSI). Combined with a virtually unlimited lexicon, it understands most of the links it comes across. An Audio Set Up Wizard automatically sets the proper microphone level on the system and the user is able to interact with it using the English recogniser without any training.

The tool uses the open architecture of Microsoft's IE 5.0 to provide all of the HTTP processing. This ensures that it supports the Java Speech API (JSAPI), allowing voice control over any Java applet employing the Java Speech API standard. The browser works with Internet Explorer (4.0 or higher) under Windows 95/98 and WNT 4.x. With Conversa Web it is not possible to use Netscape components, or to voice-enable the Netscape browser. However, it is possible to import from Internet Explorer any existing Netscape bookmarks and put them into Favourites folder named "Imported bookmarks; so they will all be voice-selectable in Conversa Web.

2.2 FreeSpeech Web

FreeSpeech Browser is a visual browser voice-enabled developed by Philips². It appears as a moveable toolbar on the PC desktop and has been designed to work only with Microsoft Internet Explorer. Actually it is free, anyone can download it from the network.

The browser is speaker-independent, the user does not have to train the system before to use it. The Audio Wizard starts automatically after installation for adjusting the audio system. It is important to get the best possible sound quality, which is needed when working with speech recognition; if there are problems when working with FreeSpeech Browser the user can run the Audio Wizard again.

FreeSpeech Browser includes Voice commands, which have been specifically created for controlling Microsoft Internet Explorer functions and surfing the Internet. The user can use the Voice command *What Can I say* to obtain a complete list of all commands. Voice commands are available for going from page to page, moving up and down in a page, opening new Explorer windows and navigating between them, updating pages, etc. FreeSpeech Browser also contains a series of commands for activating links by voice. When the link has been recognised, it will start to flash and the page will be loaded. If a link cannot be voice-activated it will be assigned a link number. Saying the commands *Number <NUMBER>*, *Go to <NUMBER>* or *Surf to <NUMBER>* (<NUMBER> stands for the actual link number) the Voice Browser opens the desired page. The user can choose between various options for assigning numbers to links: when to view link numbers (always, never, when the microphone is on), what links should receive a link number (all links, links which cannot be voice-activated).

The speech recognition engine is Free Speech 2000³, a natural speech recognition software which offers a PC voice control. It uses Philips' new HAS Technology which reduces error rates, increases speed up, constantly learns and adapts to the user's way speaking. Even if it supports a lot of languages, only the US language can be used with FreeSpeech Browser.

FreeSpeech Browser can be used on any language version of operative system and Microsoft Internet Explorer but commands, link names and link numbers can be used in English only.

² www.speech.philips.com

³ www.speech.philips.com

FreeSpeech Browser requires certain hardware and software specifications in order to function properly. It is optimised for Pentium III and MMX, it supports Windows 98 and Microsoft Internet Explorer 4.01 (with the latest Service Pack) or 5.0 as well as being Y2K compliant. Philips guarantees that the software was tested on Windows 98/Internet Explorer 4.01(with the latest Service Pack) and 5.0, but could also run under the latest releases of Windows 95 and Windows NT, based on whether the version installed supports Microsoft Active Accessibility (MSAA). They make no claims to this effect, and take no responsibility for the proper operations under these systems, nor do they offer any support for use of FreeSpeech Browser running on these operating systems.

Actually there are some conflicts between FreeSpeech Browser and Universal Serial Bus (USB) drivers, and the only solution is to uninstall the USB devices and drivers. The support staff is working to solve this problem.

2.3 Web on Call

Web-On-Call™ is a Voice Browser that allows any user with a telephone to access Internet or Intranet sites.

Developed by General Magic, Inc.⁴, is a Web server software that joins Internet and telephony networks to provide access to information residing on Web servers. By installing Web-On-Call on a Web server, users can have information via pre-recorded audio clips or Text-To-Speech synthesis, and they can browse using their phone's touch-tone keypad. The user can obtain Web pages without a computer, Internet connection or Web browsing software.

The companies need to have a Web server, regular analogic or Centrex phone lines, Dialogic telephony board or voice modems, and Web-On-Call Voice Browser software. With this configuration they no longer have to replicate and maintain the same content on multiple servers such as UNIX-based Web, OS/2-based Interactive Voice Response (IVR), DOS-based fax-on-demand, and WindowsNT-based workgroup servers. With Web-On-Call Voice Browser, Web documents are the single source of information regardless of delivery method.

The end-user must not have any training to use it and any knowledge of Internet terminology, he needs only a phone to browse.

Callers are prompted for simple touch-tone responses, and once they reach the destination information, this will be read over the phone using the Text-To-Speech synthesiser or pre-recorded voice. Text-To-Speech has an advantage over pre-recording in terms of delivering dynamically changing data on the fly. In alternative they can choose to obtain the information via fax or e-mail.

Web-On-Call Voice Browser reads information stored in Web documents and transforms the content into various delivery formats.

The user interaction and system flow can be controlled using Web-On-Call HTML control tags. These tags are added to HTML documents and convey to Web-On-Call how the HTML documents should behave and their use will not affect the Graphical User Interface (GUI) browser's handling of the page.

Install just the software and do nothing else, Web-On-Call simply will read the exact text on the HTML pages. Optionally, it's possible to add HTML tags such as "Audio

⁴ www.netphonic.com

Mute" tag to mute certain sections (such as an embedded graphic image), "Audio Insert" tag to sound bites, "Auto Play" tag to automatically start playing pages. Another option allows skipping any existing graphical links without impacting GUI browsers. For example, Web-On-Call can skip copyright information or link to an outside search engine, while GUI browsers still graphically display them.

Last minute news: General Magic is no longer supporting the product.

2.4 AudioWeb

In 1995 the Rutgers University⁵ started to work with AudioWeb project. In 1997 AudioWeb voice browser became publicly available. However a software library of tools, to make the process of creation of audible content for Web pages easier and faster, is still under development. The library is constantly updated and extended in response to the new experiences in building services.

AudioWeb allows access to web resources using a combination of speech synthesis, touch tone control and soon speech input as well: the user does not need any device. The user can access selected web sites using the telephone, and he can also create his own AudioWeb page with personal bookmarks as well as access to his personal files, email etc. The requested information is speakable to the user using Text-To-Speech software.

AudioWeb is a hyperlinked collection of audio-pages, which can be browsed and spoken to the user. The audio-pages are a generalisation on HTML pages, developed using ADML (the proposed audio-page authoring language), linked in a similar fashion as regular HTML pages and they can represent user's email, personal files, bookmarks, databases etc.

AudioWeb consists of audio server's software and hardware as well as ADML. The audio server is a machine with Windows NT platform, equipped with call processing card which handles multiple calls at the same time (from 2 to 16 calls at the same time depending on Dialogic's board model) and supports speech recognition and Text-To-Speech functionalities.

Audio-pages can be stored on any machine but in order to be read and interpreted, they have to be brought to the Audio server. A typical Audio-page is a small ASCII file, which does not contain any digital audio. Thus, an owner of an audio-page does not need to have any special equipment on his machine: she just needs to store his audio pages in his public_html directory. All processing, parsing and interpretation is carried on the audio-server.

To build an audio web site, it's possible to use some of the automated tools provided but this is not a fully automated process (this is stated by the development group). Audio page generation tools generate automatically ADML pages, i.e. audio-dB automatically creates a hierarchy of audio pages from the underlying database.

ADML enables to write an audio interface to any existing web site and have all the changes to that web site automatically reflected - through that audio interface - in what the callers to that web site are going to hear.

⁵ www.cs.rutgers.edu

2.5 speechWeb

speechWeb is an audio browser server which enables access to the Web via any phone, developed by PipeBeach,⁶ creating speech dialogues from web pages: it reads text with synthesised speech and plays pre-recorded audio. It supports multi-lingual Text-To-Speech synthesis including English, German, Italian, Dutch, French, Spanish and Swedish.

speechWeb offers the operator or service provider a quite simple way for creating applications together with third party application developers. Since the applications can be hosted anywhere on the Internet, the developers can work independently of the operator. There is no need for additional software or equipment once speechWeb has been deployed. The applications are designed in HTML and published on standard Web servers. The user will access speechWeb from any standard phone. A speech based interface can be added to the current set of value added services, centralising it to speechWeb instead of distributed among several network elements.

speechWeb supports the latest open Internet standards and protocols such as HTML 4 and ECMAScript (JavaScript). It gives the content provider and telecomm operator the freedom to create services that range from simple HTML pages to advanced interactive applications using speech recognition. The only limitation is that the pages may only contain HTML and Java scripts.

The user interacts with speechWeb and the applications through an easy-to-use interface that has the flexibility to accept both speech and touch-tone input while interacting with the user. This interface follows the desktop browser paradigms, using command like "Back" and "Forward"; important commands are reached with single key presses.

2.6 pwTelephone

pwTelephone, developed by The Productivity Works, Inc.⁷, is a product that allows telephone access to both Internet and Intranet, including access to audio components of the Web. Its philosophy transforms the phone into a Web access device to obtain information and services that are Web-based. The software runs on a standard Windows 95, 98 or NT machine and turns the machine into a telephone server.

pwTelephone architecture is a server group that consists of an extensible number of PCs each with a single input telephone line. The PCs are linked together using a simple Ethernet LAN so that a single PC on the LAN may act as the central resource for shared information and definition files. The pwTelephone server group works equally well providing information directly from its own PCs, information from servers accessible over internal Intranets, or information from the Internet itself.

pwTelephone components are a telephone interface, a telephone user interface and an Audio browser engine. The audio engine used is the one present in pwWebspeak, a non-visual browser that provides considerable flexibility in information access and auditory information presentation.

The telephone interface allows recognition of entries from the telephone keypad, while the telephone user interface provides a definable link between the telephone keypad and the commands that drive the audio engine.

⁶ www.pipebeach.com

⁷ www.prodworks.com

- The Telephone Interface recognises a phone call, hangs up, and resets the system at the end of a phone call and recognises the keypad entries made by the user.
- The Telephone User Interface provides a customisable interface so that the keypad entries may be mapped to specific pwTelephone or Audio web browser engine commands.

The Audio Browser Engine provides two useful capabilities for customisation of application and usage style: 1) the ability to define a Command menu, which may be made accessible to the user and which may provide additional navigation and functional guidance for different application uses; 2) the ability to customise the way in which contextual information on web pages, such as headings, links, or tabular information, are presented aurally to the user.

User actions are based on a PIN number. This provides the capability to have different services or paths into information. The file tyeluser.ini contains the details of how the system is presented to the specifically identified users.

2.7 WebGALAXY

WebGALAXY is a prototype system developed in 1997 by Massachusetts Institute of Technologies⁸ Laboratory of Computer Science Spoken Language System Subgroup. Due to limited resources, WebGALAXY is not available to the public at this time.

It is a conversational spoken language system for access on-line resources from within a typical browser using both speech and natural language.

WebGALAXY is an extension of GALAXY system, and now it has been integrated in GALAXY II architecture (a development of the former), the reference architecture for the DARPA Communicator program⁹. GALAXY's technology servers include SUMMIT for speech recognition, TINA for language understanding, and GENESIS for language generation.

WebGALAXY architecture consists of two parts: a hub and a standard Web browser. The hub maintains the state of current discourse with the user, mediating the information flow between the various servers and the web browser. The Web browser is used to provide all graphical user interfaces to WebGALAXY.

To start WebGALAXY, the user brings up the WebGALAXY home page within a Web browser (such as Netscape Navigator or Microsoft Internet Explorer), selects either the Java or forms interfaces, enters a phone number if spoken interaction is desired, and clicks the Start button. Users who enter a phone number will be shortly called for a telephone-based conversation. In any case, the user can interact with the system at any time through keyboard entries and mouse clicks.

As domain server, WebGALAXY stores information in a relational database. When a user asks a question over the telephone, WebGALAXY uses SUMMIT to convert the spoken sentence into text, TINA to parse the text into a semantic frame¹⁰ and GENESIS to build a Standard Query Language (SQL) query for the database using the semantic frame's basic terms. At this point WebGALAXY executes the SQL query

⁸ www.sls.lcs.mit.edu

⁹ The DARPA Communicator program supports the creation of intelligent conversational interfaces to distributed information. For a detailed description of the Communicator Program see <http://fofoca.mitre.org>

¹⁰ A semantic frame is a grammatical structure containing the basic terms needed to query the database.

and retrieves the requested information from the database, while TINA and GENESIS convert the query result into a natural language sentence so the system delivers the generated sentence to the user via voice (using a speech synthesiser) and/or display

Depending on user specification, GENESIS may access Italian, English, or Japanese language tables during language understanding procedures.

2.8 Speech Aware Multimedia

Speech Aware Multimedia (SAM), developed by Texas Instruments¹¹ Media Technologies Laboratory, is a speech interface to the Web that allows access to information using voice. SAM is not only useful to browse the web via voice, but for the development of Web-based speech application too.

This tool is speaker independent and it offers a continuous speech and an arbitrary vocabulary recognition system. Speakable commands allow the user to browse the Web, speakable bookmarks to retrieve pages, speakable links to select a link by simply speaking it, and smart pages to do natural speakable queries. SAM has the ability to incorporate new grammars and vocabularies "on the fly" and to modify grammars dynamically in the recogniser. Speakable commands help the user to display pages, to scroll up and down, to go back or forward, to add pages to the speakable bookmarks. The user can customise the phrases used for the commands to have more convenient expressions to interact with.

Speakable links enable the user to pronounce the link. In fact SAM parses the HTML page to determine the links and the URLs associated with them and then it transforms the string of words into a grammar "on the fly" for the recogniser.

Smart pages are Web Pages that contain a link to a grammar appropriate for a page or for a set of a page. The web page author can define his page specific language and interpret the result in an application specific manner. When SAM loads a smart page, it incorporates the associated grammar in the current grammar set; after the user makes a query, SAM sends back words that will be interpreted by the page to provide an answer.

All tags behind images are treated just like speakable links; framed pages and Java programs are voice supported.

During the voice interaction the user can mix keyboard and mouse input, and the mouse can be controlled by voice. If the page retrieval becomes too slow, the user can vocally interrupt the loading; finally, by using the commands "Go to sleep" or "Wake up" he/she can do totally hands-free operation.

Another feature of SAM is an Application Programming Interface (API) that allows Java authors to speech enable their Java applets by specifying grammars and actions appropriate for various contexts. In this way an applet or an application can talk to the speech recogniser, in fact when operating with SAM an applet may request the speech focus to behave as it were operating in a stand-alone application. Given the SAM system, users can create meaningful speech applications by using any one of several GUI-based web page creation tools to create pages.

¹¹ www.ti.com

2.9 SpeechHTML

SpeechHTML is a product developed by Vocalis¹² that enables people to access and obtain audio information from any Web site, written in standard HTML, interacting with the Internet by telephone and spoken commands.

SpeechHTML allows companies to offer their own interactive telephone services purchasing a Web site and a SpeechHTML license. Subscribers give Vocalis their web address, register their sites on SpeechHTML's own administration Web site, and get back a specific telephone number that will connect callers to the Web site.

SpeechHTML integrates a gateway package, called SpeechHTML Gateway, with speech recognition and text to speech conversion to allow people to speak to a Web site and get the information they need. The conversion server, reads HTML tags and translates expected mouse clicks into expected voice response, whether spoken or using touch tones

When a customer calls the appropriate number, this call connects him/her to the SpeechHTML gateway, which creates a link to the specific Web site. Data at the site, encoded in standard HTML, is accessed and retrieved. When SpeechHTML accesses a web page, it either interprets or ignores HTML tags. The HTML tags influence the way the Web site is read out. For example, general text is simply passed to the text to speech system that converts it to speech. The text to speech element, which is sent down the telephone to provide the information required by the caller. If a page has hyperlinks to further information, these will be presented as choices to the caller, the speech recognition engine will interpret what the caller asks for and the new web page will be accessed and read out. If a page contains a form, the caller will be prompted for the information to fill in the gaps. When callers choose from a list of options, answers are recognised from the advanced speech recognition engine; free-form answers are recorded.

SpeechHTML works with web pages written in HTML v3. If web pages are written with using a later version, some of the extensions may not be supported and it will ignore tags it does not recognise. The SpeechHTML Web site contains information to help the developing of effective services that work both graphically using a web browser, and over the telephone.

Vocalis gives the developers some guidelines on creating effective dialogues, some hints on how to get the best out of the text to speech system, and some example Web sites to copy for performing specific tasks. The data can be changed and updated simply by making changes to web pages.

The speech recognition in SpeechHTML uses:

Word spotting to recognise when callers use extra words other than those that are expected. For example, if SpeechHTML is expecting to hear the name of a football club so that it can play out the required score, the caller could say "Juventus", "Juventus, please", or "Tell me the Juventus score".

Talkover to interrupt the prompt by saying the next option. This is useful for experienced or frequent callers who know their way around the service and want to get to information quickly.

Out of vocabulary rejection to repeat a question if it does not understand the caller's response, creating more natural dialogues.

¹² www.vocalis.com

The SpeechHTML system uses text to speech synthesis to read pages out to the caller.

It lets developer makes changes to Web sites that are immediately heard by the callers. The *User Configurable Dictionary* allows developers to add words to the text to speech vocabulary that may not be said exactly as they are written. This is useful for company or product names such as SpeechHTML. The *user configurable dictionary* contains pronunciations of words that are specific to the service.

If a web page is complex and takes a while to download, SpeechHTML plays a message to the caller. The developer can change the message played while the caller waits for the page to download. You can change

SpeechHTML runs under SCO's Open Server operating system, and the two companies have formed joint marketing programs for the two products. SpeechHTML will be available both for licensing by end users, and as a system that equipment vendors and service operators can buy to incorporate as a product.

3 Application development environments

3.1 VoxML

Motorola offers a development environment to build voice applications that will be browse using voice. The environment includes:

- VoxML Development Platform;
- VoxML Software Development Kit.

3.1.1 VoxML development platform

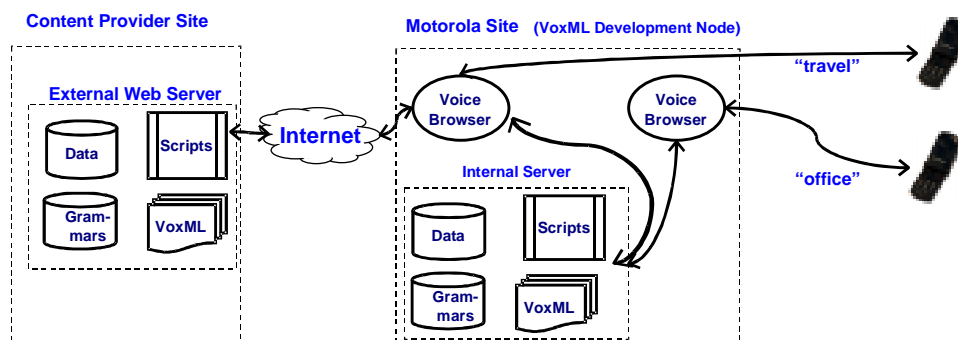


Figure 1 - VoxML development node

The two major components of the platform are the *external web server* at the content provider/partner site and the *VoxML development node* at Motorola. The first one is used for initial development and testing of the VoxML application occur, the second one is available to test the application further using the same voice technology as the commercial platform.

As shown in the Figure 1 the VoxML development node contains the following:

- Voice Browsers: the software component that fetches and interprets VoxML dialogs and holds a conversation with the user using the speech recognition and synthesis components.
- Voice Recognition Hardware/Software: the node and the commercial platform share a speaker-independent, large-vocabulary, continuous-speech recognition engine.
- Text-to-Speech Hardware/Software: the node can use a variety of speech synthesis (text-to-speech) packages.
- Pronunciation Engine: in the case of option list inputs, the voice browser needs to generate phonetic representations of the words to be recognised. The pronunciation engine is used to generate several (currently up to 5) pronunciations for each proposed word or short phrase.
- Grammars: VoxML includes a standard set or a user-defined set of input grammars (numbers, digits, times, dates, etc.) which are pre-compiled and stored on the development node.
- Telephony Interface: an interface between the phone lines from the commercial system and the VoxML development node.

Apart from using voice and audio for the user interface, the voice browser behaves much like other Web browsers. It fetches data over the Internet using the Web's

standard URL addressing scheme and HTTP protocol, stores "cookies" on behalf of the user, and caches frequently accessed pages.

The voice browser recognises the voice request and translates it into a URL for a service provider's Web server. The Web server processes the request as if it were a request from a Web browser and responds with a VoxML™ "page". The voice browser interprets this VoxML™ page to relay information to the user via audio voice. The voice transaction would continue until the user had received all of the information necessary and would complete the interaction by ending the call.

3.1.2 VoxML Software Development Kit

Motorola VoxML 1.1 SDK is available for downloading at VoxML Web site¹³, and it requires several software products: Windows NT 4.0 (Service Pack 3) or Windows 95, Microsoft IE 4.0.1 or later, Microsoft Internet Client SDK, Microsoft Agent Components Version 1.5, Lernout & Hauspie True Voice Text-To-Speech Engine Version 1.5, Microsoft Command and Control Speech Engine Version 3.0, Web Server Software that supports HTTP 1.1 and server-side scripting.

VoxML voice applications are developed using the desktop software development kit, which enclose the VoxML markup language, the VoxML Simulator and the VoxML Viewer.

VoxML markup language has a structure designed for voice applications and is based on the Extensible Markup Language (XML), a W3C standard. It follows the syntactic rules of XML, with semantics that support the creation of interactive speech applications. VoxML offers the same building blocks as HTML: text (which is read via text-to-speech), recorded sound samples (analogous to images), navigational controls, and input controls. VoxML is designed and delivered in whole-dialog units, with dialogs presented in much smaller units, called *steps*, while HTML is designed and displayed in whole-page units.

A VoxML document has a hierarchical structure, in which every element (except the *dialog* element) is contained by another element. There is a strict containment relationship enforced by the language, to avoid ambiguity about which element contains which.

The basic VoxML elements are *dialog*, *step*, *prompt*, *help*, and *input*. The two most fundamental elements are the *dialog* element and the *step* element, which provide the basic structure of a VoxML application.

The *dialog* element defines the scope of a VoxML document, and it contains all other VoxML elements (one can equate a *dialog* to a file).

The *step* element defines a state within a *dialog*, or to say it another way, the *step* element defines an application state. *Step* elements in a VoxML application are executed based on the user's responses not on the order of the *steps* within the source file: the order in which they are defined has no impact on the order in which the voice browser navigates through them.

Together, the *dialog* element and the associated *step* elements define a state machine that represents an interactive dialogue between the application and a user. When the

¹³ www.VoxML.com

VoxML voice browser interprets the VoxML document, it navigates through the DIALOG to different *steps* as a result of the user's responses.

The *prompt* element defines the content that is to be presented to a user in a *step*. This content consists of intermixed text and audio elements and it is read via text-to-speech technology.

The *help* defines the behaviour of the VoxML application when the user asks for help. If the application developer does not define the behaviour of *help* for a given *step*, the system default behaviour will be used. The default behaviour for the *help* element is to stop the *prompt* (if one is playing), speak the phrase "No help is available", remain in the current *step*, and process any interactive *input*.

The *input* element is used to define the valid user input in a *step*. The application developer can define the type of input (Date, Digits, Grammar, Hidden, Money, None, Number, Phone, Profile, Time, record, etc) as well as specific values that are to be recognised.

As when any VoxML document is interpreted, the voice browser begins by executing the *step* called "init". The user will hear the text contained by the *prompt* element. If the user were to ask for "help" before making a selection, the user would hear the text contained with the *help* element. After the user makes a selection, the voice browser will execute the *step* named "confirm", which will simply read back the user's selection and then exit the application.

The developer writes the VoxML application using a common text editor, the VoxML Viewer, which allows him to verify the structure of a VoxML dialog and to parse the dialog to produce a graphical representation of the corresponding XML tree structure.

The main role is played by the VoxML simulator, which provides an interactive method for executing and testing a VoxML based application. It has a dialog window interface that allows control of an application's execution. A Microsoft Agent character is used with the VoxML Simulator to speak the prompts corresponding to the current dialog of the voice application. The VoxML Simulator dialog window include:

- an area for specifying the URL of the VoxML application;
- a text area display of the last prompt (if applicable, this prompt is spoken by the Microsoft Agent character),
- input box to type in a response to a prompt,
- button labelled "speak" sends the typed response to the application,
- when applicable, a pop up menu of valid responses for the current step in the dialog,
- error generation section contains a pop-up menu of available error types and a button labelled "generate" to simulate the error,
- debug log section displays any debug information along with "save log" and "clear log" buttons,
- button labelled "show source" shows or hides the source code for the current dialog, and
- button labelled "quit" exits the simulator.

3.2 Mobile Application development Kit

The Mobile ADK developed by Motorola unites development tools for both WAP (Wireless Application Protocol) and VoxML™ applications in a single toolkit. The current version of the VoxML™ simulator in conjunction with various WAP-compliant modules, synthesises the source code and builds onto existing components to create a single tool that is capable of testing both the WML (Wireless Markup Language) and VoxML™ applications. It includes all of the functionality found in the individual VoxML™ 1.1 SDK (cf. 3.1) and WAP 1.1 specifications, plus an additional option, which allows users to switch between the voice and data modes on the fly. The look and feel of the simulator is a hybrid of the current simulators used by the different SDKs.

The product includes a WAP simulator tool, which displays a Motorola WAP-phone image with simulated (WML and VoxML™) data being piped onto the phone display, along with a window (called as Transcript window) for tracing the application behaviour. During the voice mode, the simulator invokes speech engines and agent components to display text, while enabling the users to interact via voice.

The Mobile ADK Simulator provides an interactive method for executing and testing the Mobile ADK application scripting. It has a dialog window interface that allows control of an application's execution. A Microsoft Agent Character is used with the Mobile ADK Simulator to speak the prompts corresponding to the current dialog of the voice application.

The software needed to run Mobile ADK is based on Windows NT 4.0 / 95 operating system, with Microsoft Internet Explorer 4.0 or higher, Microsoft Java Virtual Machine ver 3.2 and Microsoft Agent Components (including ASR and TTS engines).

3.3 VoiceXML

VoiceXML Forum, an industry organization founded by AT&T, Lucent Technologies, and Motorola, promotes the VoiceXML language. The forum is positioned to make the resource of World Wide Web voice- and phone-enabled; the markup language is proposed as a computer language to create Web content and services that can be accessed by voice and phone.

VoiceXML Forum main goal is to bring the full power of web development and content delivery to voice response applications, and to free the authors of such applications from low-level programming and resource management. In this way the Forum gives a VoiceXML language description only and not a formal specification.

3.3.1 VoiceML architectural model

The architectural model which supports VoiceXML application development is depicted in Figure 2.

The Document Server (hitherto a web server) maintains overall service logic, perform database and legacy system operations, and produce dialogs. It processes request from a client application (called Interpreter Context) and produces VoiceXML documents in reply. A VoiceXML document describes the site specific dialog and directs access to information on the site.

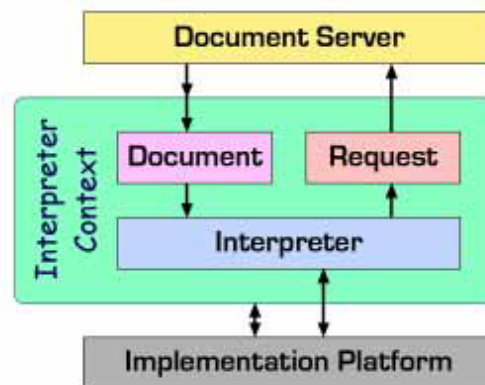


Figure 2 - VoiceXML architectural model

The Interpreter Context is responsible for call control services and for acquiring the initial VoiceXML document. It contains the VoiceXML Interpreter that leads the interaction dialog (specified in the VoiceXML document) after answer. The Interpreter Context is expected to acquire documents for the VoiceXML Interpreter to act on. The document request is generated by the interpretation of a VoiceXML document, while other requests are generated by the interpreter context in response to events outside the scope of the language, for example detecting an incoming phone call, answering the call, etc. The two modules control the Implementation Platform.

The Implementation Platform generates events in response to user actions (e.g., spoken or character input received, disconnect) and system events (e.g., timer expiration). Some of these events are acted upon by the VoiceXML Interpreter itself, as specified by the VoiceXML Document, while others are acted upon by the VoiceXML interpreter context. The Implementation Platform must have requirements about audio input and output. It can provide audio output using audio files and/or using text-to-speech, and must be able to intermix the two types of audio freely (audio files are referred to by a Universal Resources Identifier, URI). It is required to detect and report character and/or spoken input, entered by the user, simultaneously and to control input detection interval duration with a timer whose length is specified by a VoiceXML document.

With this architecture a Voice Service can be seen as a sequence of interaction dialogs between a user and an implementation platform, where the dialogs are provided by document servers (internal or external to the implementation platform). User input affects dialog interpretation and it is collected into requests submitted to a document server which may reply with another VoiceXML document to continue the user's session with other dialogs.

3.3.2 VoiceXML markup language

VoiceXML markup language derives its lexical and syntactic conventions from XML¹⁴.

The main goals of VoiceXML markup language can be summarised as:

- Minimal client/server interactions.
- Separation between low-level, and platform-specific details.

¹⁴ XML Reference Manual www.xml.com/axml/testaxml.htm

- - Service portability across implementation platforms.
- - Safely handling of shared network-based applications.
- - Easy to use for simple interactions, and yet extensible for complex ones.
-

To satisfy these goals the markup language:

- - supports ease of authoring for common types of interactions;
 - - accommodates platform diversity in supported audio file formats, speech grammar formats, and URI schemes;
 - - has a well-defined semantics that preserves the author's intent regarding the behaviour of interactions with the user;
- enables a separation of service logic from interaction behaviour;
 - does not include complex computational capabilities, database operations, or legacy system operations,
 - provides ways to link documents using URIs, and also to submit data to server scripts using URIs,
 - provides ways to identify exactly which data to submit to the server, and which HTTP method to use in the submittal,
 - supports the tracking of information presented by the interpreter.

The language provides means for collecting character and/or spoken input, assigning the input to document-defined request variables, and making decisions that affect the interpretation of documents written in the language. A document may be linked to other documents through URIs. When a link is followed, request variables and their values, if present, are submitted to the link's URI.

A VoiceXML *document* (or a set of documents, called an *application*) forms a conversational finite state machine. The user is always in one conversational state, or *dialog*, at a time. Each dialog determines the next one to transition to. *Transitions* are specified using URIs, which determine the next document and dialog to use. If a URI does not mention a document, the current document is assumed. If it does not mention a dialog, the first dialog in the document is assumed. A conversation is terminated when a dialog does not specify a successor, or if it has an element that explicitly exits the conversation.

An *application* is a set of documents sharing the same *application root document*. Whenever the user interacts with a document in an application, its root document is also loaded. The root document remains loaded while the user is transitioning between other application documents, and it is unloaded when the user transitions to a document that is not in the application. While it is loaded, the application root document's variables are available to the other documents as *application variables*, and its grammars can also be set to remain active for the duration of the application.

A *session* begins when the user starts to interact with a VoiceXML interpreter context, continues as documents are loaded and processed, and ends when requested by the user, a document, or the interpreter context.

A VoiceXML document is primarily composed of top level elements dialogs. There are two kinds of dialogs: *form* and *menu*.

Forms define an interaction that collects values for a set of form variables. If a form-level grammar is present, it can be used to fill several fields from one utterance.

Menu presents the user with a choice of options and then transitions to another dialog based on that choice.

Each dialog has one or more speech and/or DTMF (touch-tone) *grammars* associated with it. In simple *computer directed*¹⁵ applications, each dialog's grammars are active only when the user is in that dialog. In *mixed initiative*¹⁶ applications, where the user and the computer alternate in determining what to do next, some of the dialogs are flagged to make their grammars *active* (i.e., listened for) even when the user is in another dialog in the same document, or on another loaded document in the same application. In this situation, if the user says something matching another dialog's active grammars, execution transitions to that other dialog, with the user's utterance treated as if it were said in that dialog. Mixed initiative adds a surprising amount of flexibility and power to voice applications.

A *link* is a special construct supporting mixed initiative: it appears at the top level of a document and specifies a bundle of grammars to listen for whenever the user is in that document (or that document's application). If the user says or keys in something matching a link, the link's destination URI is transitioned to. A link is thus a *transition pattern* that applies to all the dialogs in its scope, a much terser and more efficient alternative to specifying the same transition repeatedly.

Events are thrown by the platform when the user does not speak, doesn't speak intelligibly, asks for help, etc. The interpreter also throws events if it finds a semantic error in a VoiceXML document. Each element in which an event can occur has an event handler. Each event handler is formed from any catch elements (and syntactic shorthands for catch elements) enclosed by that element. An event handler also inherits catch elements ("as if by copy") from each of its higher level elements, as needed. In this way, common event handling behaviour can be specified at any level, and it applies to all lower level event handlers.

3.4 SpeechML

IBM has developed a package to build network-based conversational applications. The package includes Speech Markup Language and the conversational browser that implements the markup language.

A conversational application developed using SpeechML is an application that interacts with the user by voice input and voice output. A network-based application refers to one in which the elements of the conversation that define spoken output and input - pages of Speech Markup Language - may be obtained over the network. The term "application" in this context should be understood in the same broad sense in which a set of related Web pages containing text, links, and forms, together with the code on a server which processes the forms, constitute an application.

3.4.1 SpeechML architectural model

The architectural model proposed is depicted in Figure 3.

¹⁵ A computer directed application is an application where the computer directs the entire conversation

¹⁶ A *mixed initiative* application is an application where both the computer and the human direct the conversation.

A conversational browser runs on up to a client device (e.g. a desktop PC, a PDA, an automobile computer, a smart telephone, or a "dumb" telephone¹⁷). On the same device there could be an application running locally which could use the same speech markup language to communicate with the browser.

Different kinds of services run on up to different servers connected to the client device over the Internet and using a speech markup language to communicate with the client (in the Figure 3 three different servers are depicted).

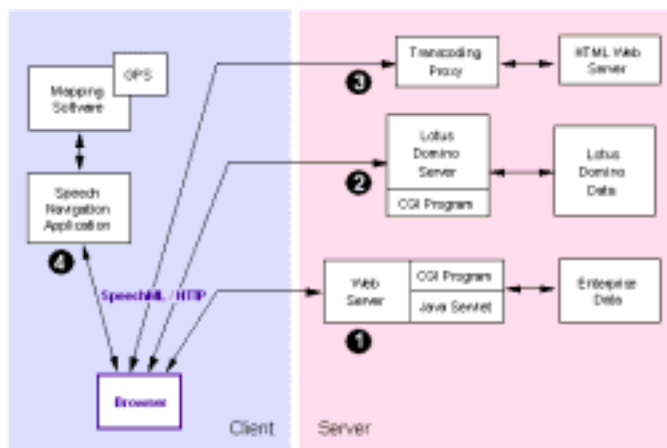


Figure 3 - SpeechML architectural model

3.4.2 SpeechML markup language

Speech Markup Language is an XML application, meaning that it is defined as a set of XML tags or elements.

The notion of visiting a Speech Markup Language element is equivalent of visiting an URL. Visiting an element is an action taken by the browser, typically in response to a spoken user input (although it may also be in response to some other kind of user input or some other kind of asynchronous event). Visiting a Speech Markup Language element will cause the browser to produce spoken output, depending on the type and content of the Speech Markup Language element - for example a body element is read, the choices of a menu element are listed, and so on. Visiting an element also affects the set of spoken responses that the browser will accept from the user and the manner in which this happens depends on the element itself.

The primary SpeechML elements are *page*, *body*, *menu*, and *form*.

A *page* element is a group of related Speech Markup Language elements. It is used as a container to group other Speech Markup Language elements such as menus, bodies, and forms (it doesn't contribute to the conversation between the application and the user). The top-level element of a Speech Markup Language document is a page; pages may also contain nested pages.

A *menu* element presents the user with a set of choices. It has a similar function for Speech Markup Language as hypertext links and menus have for HTML. Each choice is represented as a *choice* element, with a target attribute that is a URL that the browser will visit if the user selects that choice. When a menu is visited, its

¹⁷ A "dumb" telephone functions as a microphone at the end of a long cord, which needs to be connected to a browser at the other end of the phone line

introductory text is read, followed by the prompt text of the choices. The user may select one of those choices by saying an attention word or phrase, followed by one or more significant words from the prompt text of the choice in the same order in which they occur in the prompt. When the user selects one of the choices, the browser visits the URL specified by the target attribute of the choice, which may be either an absolute URL or a URL relative to the URL of the Speech Markup Language document that contains the menu.

A *body* element contains a chunk of text to be spoken to the user. For information-retrieval applications (such as e-mail and news) a body element will typically be used to contain the information that the user is seeking. A body element may be plain text, or it may mark up to improve the text-to-speech synthesis using the Java Speech Markup Language (JSML). Since a body typically contains only text to be spoken, it usually offers no clue as to what the user can do next. This problem is solved by the next attribute of the body tag. The value of the next attribute is a URL specifying another Speech Markup Language element - such as a menu or a form - to be visited after the body has been read to the user.

A *form* element collects one or more pieces of information, or fields, from the user. It is typically used for collecting information, such as names, addresses, numbers, and so on, that would be impractical to present as a list of choices in a menu. A form has some introductory text, followed by a list of fields. Each field has a prompt and a reference to a grammar in Java Speech Grammar Format (JSGF) specifying the set of possible values for the field. When the form is visited, the introductory prompt is read and the user is prompted to supply the values of the fields. The user may fill in the value of any field - either when prompted or at any other point as long as the form is still in scope - by saying a portion of the prompt followed by a legal value of the field.

A *body*, *menu*, or *form* element may have a Name attribute, which allows the element to be visited even if it isn't the first element on a page, by using the name as the reference part of a URL in the standard *fragment URL* reference format.

A Speech Markup Language application designer can control the duration for which the spoken input associated with a menu or form is active, using the *scope* attribute. For example, it's very useful to have main menus or major topic menus active for the entire sessions so that users can jump directly to the choices provided by the menus without first having to navigate back through a "maze" of menus. However, in some circumstances this can decrease speech recognition accuracy and produce surprising results.

3.4.3 The conversational browser

The package contains a conversational browser whose features are not intrinsic to SpeechML but rather are characteristic of this browser implementation. The browser implements a number of built-in commands, similar to the built-in functions presented on menus and toolbars in visual browsers. The user can ask to *quit* or *shut up* the browser, obtaining the stop of current spoken output and the wait for further instructions. Using the *say again* or *repeat* commands, the browser repeats the most recently visited element (menu, form or body). With the *go back* or *go forward* commands the browser goes back to the previously visited element, or goes forward to the next visited element in the history list. Using the *beginning* command the browser goes back to the first element visited (the home page). A feature of the browser applied to all spoken input is that the user may interrupt it at any time (this is

sometimes called barge-in). This is true at any point where a spoken command is allowed, such as a menu item selection.

The browser addresses implements a "say what you hear" approach, in which acceptable inputs always echo the spoken prompts that are presented to the user. As mentioned previously, visiting a Speech Markup Language element changes the set of user inputs that the browser will accept. Following the "say what you hear" approach, the inputs accepted are in general echoes of the menu or form prompts, or some abbreviated version of the prompt.

Generally a Speech Markup Language browser offers two enhancements to this idea that put the user, rather than the browser, in control of the conversation:

- Say what you heard: every menu or form that the user has heard in the past remains active thereafter. So for example, when he hears on the main menu choices for "e-mail" and "news", he might immediately select the e-mail application by saying "select e-mail". Then, at any later time, he might recall that you heard "news" on the main menu, and choose to go to the news application by saying "go to news", without having to navigate back to the main menu. Or, if he has forgotten all the selections that were on the main menu, he might go back to the main menu by saying "go to main menu". The same principle applies to every menu and form he has heard.
- Say what you will hear: not only menus and forms remain active throughout the browser session, they in fact become active when they are loaded by the browser, before they've even been visited. This is a feature for experienced users of a Speech Markup Language application, who have used the application enough to anticipate certain menus or forms, such as the menu that offers a set of options for disposing of an e-mail message after it has been read.

3.5 Java Speech API

The Java Speech API¹⁸ was developed by Sun Microsystems, Inc. in collaboration with speech technology companies: Apple Computer, Inc., AT&T, Dragon Systems, Inc., IBM Corporation, Novell, Inc., Philips Speech Processing, and Texas Instruments Incorporated.

The JavaTM Speech API is a part of a family of JavaTM Media APIs that work together as a suite to provide customers with enhanced graphics and extended communications capabilities.

The JavaTM Speech API allows developers to incorporate speech technology into user interfaces for their Java programming language applets and applications. This API specifies a cross-platform interface to support command and control recognisers, dictation systems and speech synthesisers. The API can be divided into three relatively self-contained areas: resource management, speech recognition and speech synthesis.

The resource management provides audio connectivity and supports the selection and management of speech recognition and speech synthesis engines.

About the other two areas Sun has developed the Java Speech Grammar (JSGF) to provide cross-platform control of speech recognisers and the Java Speech Markup Language (JSML) to provide cross-platform control of speech synthesisers.

¹⁸ JavaSpeech API 1.0 specification is available at <http://java.sun.com/products/java-media/speech>

The Java Speech API is a freely available specification and so anyone can develop an implementation. Actually the following implementation are known to exist:

- *IBM's "Speech for Java"*: Implementation based on IBM's ViaVoice product, which supports continuous dictation, command and control and speech synthesis. It supports all the European language versions of ViaVoice (US & UK English, French, German, Italian, Spanish and Japanese).
- *IBM's "Speech for Java" on Linux*: Beta version of "Speech for Java" on Linux. Currently only supports speech recognition.
- *Lernout & Hauspie's TTS for JSAPI*: Implementations based upon ASR1600 and TTS3000 engines, which support command and control and speech synthesis. Supports 10 different voices and associated whispering voices for the English language. Provides control for pitch, pitch range, speaking rate, and volume.
- *Conversa Web 3.0*: Conversa Web is a voice-enabled Web browser that provides a range of facilities for voice-navigation of the web by speech recognition and text-to-speech. The developers of Conversa Web chose to write a JSAPI implementation for the speech support.
- *Festival* : Festival is a general multi-lingual speech synthesis system developed by the Centre for Speech Technology Research at the University of Edinburgh. It offers a full text to speech system with various APIs, as well an environment for development and research of speech synthesis techniques. It is written in C++ with a Scheme-based command interpreter for general control and provides a binding to the JSAPI. Supports the English (British and American), Spanish and Welsh languages.

3.5.1 Java Speech Grammar Format

The JSGF¹⁹ (Java Speech Grammar Format) defines a platform-independent, vendor-independent way of describing one type of grammar, a rule grammar. It uses a textual representation that is readable and editable by both developers and computers, and can be included in Java source code.

A *grammar* is composed of a set of rules that together define what may be spoken. Rules are combinations of speakable text and references to other rules, and each rule has a unique rulename. Although rulenames are unique within a grammar, separate grammars may reuse the same simple rulename. Qualified names and fully-qualified names are used to reference between grammars without ambiguity and the local rules have precedence on the others.

A *rule* grammar specifies the types of utterances a user might say (a spoken utterance is similar to a written sentence). For example, a simple window control grammar might listen for "open a file", "close the window", and similar commands. What the user can say depends upon the context. Applications know the context, so applications are responsible for providing a speech recogniser with appropriate grammars.

The definition grammar contains two parts: the grammar *header* and the grammar *body*. The grammar *header* includes a self-identifying header, declares the name of the grammar and declares imports of rules from other grammars. The grammar *body*

¹⁹ Java Speech Grammar specification is available at <http://java.sun.com/products/java-media/speech>

defines the rules of the grammar, some of which may be public: the order used to define the rules is not significant.

A *token* is the part of a grammar that defines what may be spoken by a user. Most often, a token is equivalent to a word even if it might be a sequence of words or a symbol. A token represents a reference to an entry in a recogniser's vocabulary, often referred to as the lexicon. The recogniser's vocabulary defines the pronunciation of the tokens: with the pronunciation, the recogniser is able to listen for that token.

The Java Speech Grammar Format allows multi-lingual grammars, that is, grammars that include tokens from more than one language. However, most recognisers operate mono-lingually so a typical grammar will contain only one language. The application has the responsibility to load the right grammar into a recogniser and to ensure that it has appropriate language support.

3.5.2 Java Speech Markup Language

The Java Synthesis Markup Language (JSML)²⁰, an SGML-based markup language, is being specified for formatting text input to speech synthesisers. The language elements provide a speech synthesiser with detailed information on how to say the text. JSML includes elements that describe the structure of a document (paragraphs and sentences), provide pronunciations of words and phrases, and place markers in the text. Further prosodic elements control phrasing, emphasis (stressing or accenting), pitch, speaking rate, and other important characteristics of a speech synthesiser, improving naturalness and understandability. It uses the Unicode character set, so JSML can be used to mark up text in most languages of the world. Pronunciations can be specified for words, phrases, acronyms and abbreviations to ensure understandability. Explicit control of pauses, boundaries and emphasis can be provided to improve naturalness.

JSML has been developed to support as many types of applications, and to support text markup in many different languages. To make this possible, JSML marks general information about the text and, whenever possible, uses cross-language properties. Although JSML may be used for text in Japanese, Spanish, Tamil, Thai, English, and nearly all modern languages, a single JSML document should contain text for only a single language. The applications manage and control the speech synthesisers if output of multiple languages is required.

The application is always responsible for converting the source information to JSML text using any special knowledge it has about the content and format of the source information. For example, an email application can provide the ability to read email messages aloud by converting messages to JSML. This means the conversion of email header information (sender, subject, date, etc.) to a speakable form and a special processing of text in the body of the message (for handling attachments, indented text, special abbreviations, etc.).

JSML is a subset of XML, and so it has a standardised, extensible syntax that is not tied to the Java Speech API (JSAPI). JSML syntax consists of *structural*, *production*, and *miscellaneous* elements.

²⁰ The beta version of the JSML specification is available at <http://java.sun.com/products/java-media/speech>

Structural elements are *para*, which declares a range of text to be a paragraph; and *sent*, which declares a range of text to be a sentence.

Production elements are *sayas*, which defines how to pronounce a word or a short phrase; *emp*, which specifies a level of emphasis for a word or for a text; *break*, which marks a break in the speech; *pros*, which specifies prosodic information for a text.

Miscellaneous elements are *marker*, which requests a notification from the speech synthesiser to the application when it is reached during the synthesiser's production of audio for a text; *engine*, which provides information to use a synthesiser's special capability.

4 Research

The scientific community is very intercede about voice browsers. The following is a description of W3C Voice Browsers working group activity.

4.1 W3C Voice Browsers activity

W3C held a workshop on "Voice Browsers" in October 1998. The participants were people involved in developing voice browsers for accessing Web based services. The workshop concluded that the W3C would have organised a working group to bring together interested parties to collaborate on the development of joint specifications for voice browsers, particularly since these efforts concern subsetting or extending some of the core W3C technologies, for example HTML and cascade Style Sheet (CSS). As a response, a briefing package has been written to establish a W3C "Voice Browser" Activity and Working Group as a first step.

The Working Group has the mission:

1. To prepare and review documents related to Voice Browsers, for instance, relating to dialog management, extensions to existing Web standards, speech grammar formats and authoring guidelines.
2. To serve as a coordinative body with existing industry groups working on related specifications.
3. To serve as a pool of experts on Voice Browsers, some of which will participate in the other W3C working groups relevant to Voice Browsers.

5 V-commerce

Leading technology software, infrastructure and services companies are collaborating with Nuance to form the V-Commerce Alliance²¹. Typical V-Commerce applications will make use of products and services in several categories including e-commerce and application servers, packaged applications, telephony hardware and software, systems integration and Nuance's natural language speech recognition. These companies are championing a consistent set of standards for voice-enabling web applications using natural-language speech technology from leading speech recognition vendors like Nuance Communications.

V-Commerce extends the power, ease of use and sophistication of electronic commerce applications to people. This is accomplished through the capabilities of natural language speech recognition that enables anyone around the world to use their voice to execute transactions. The combination of Nuance's speech recognition engine with the technologies provided by members of the V-Commerce Alliance provides a platform on which businesses can simultaneously service their Web and phone-based customers.

The traditional electronic commerce is only available to people who have access to a computer. V-Commerce enhances electronic commerce by making it more accessible, easy to use, affordable and secure than the e-commerce. In fact the transactions can be executed using the device that's most convenient, the users simply say what they want without having to learn to use new technologies or devices, V-commerce is available to everyone that has a telephone, and a person's unique voice print protects access to secure information

There are three typical usage scenarios of implementations of a V-Commerce application:

- *Speech In, Speech Out*: these applications typically involve a dialogue over the phone with a speech system, making them well suited to transactions that would typically take place today using a touch-tone keypad, or by speaking to an agent in a call center.
- *Common Business Server*: all e-commerce applications are driven by a combination of application logic, business logic and database content. These applications commonly have a wealth of understanding and intelligence about the transaction to be executed, the conditions under which they may be executed, and the people who are entitled to execute them. A goal of V-Commerce is to leverage this existing knowledge base and application effort, and significantly broaden the range of possible users of current systems.
- *Speech In, Web Out* : these applications will allow that a person speaking over a phone, or through a microphone attached to a desktop or some other device, will have the output displayed in a Web browser or on a thin client device such as a PDA or pager.

5.1 V-commerce development environment

The V-Commerce development environment is based on component, communication, user interface, and middleware standards. Using these standards, speech applications can be easily integrated with Internet application and database servers.

²¹ www.v-commerce.com

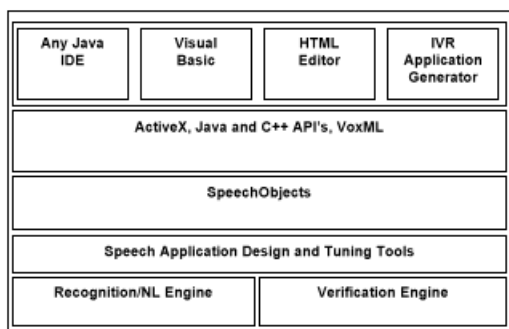


Figure 4 - V-Commerce development environment

In Figure 4 the V-commerce development environment.

At the foundation of the V-Commerce development environment there are the speech recognition, natural language, and speaker verification engines provided by Nuance. Nuance's speech application design and tuning tools incorporate the knowledge and understanding embodied in highly accurate speech recognition applications. These tools have the ability to design SpeechObjects, reusable speech components that are based on open industry standards such as ActiveX and Java. SpeechObjects can be customised, assembled or created in a variety of graphical development environments including any Java development environment, Visual Basic or application generator on an IVR platform. They are designed to be portable so that applications built with them can be deployed on major server and IVR platforms. Further they do not require developers to have specific speech or linguistics experience, they significantly decrease the level of expertise needed for developing high-quality speech applications. The development infrastructure enables companies that are currently leveraging the power of the web for e-commerce, to easily extend their applications to include a speech interface.

6 Portals

A portal is intended to be an all-in-one entrance to the Internet. Not merely a search site, but a site that provides you with Internet services: email, chat rooms, free personal WebPages, shopping, guides, etc. The first Web portals were online services, such as America On Line (AOL), that provided access to the Web, but by now most of the traditional search engines have transformed themselves into Web portals to attract and keep a larger audience. Yahoo!²² and Excite!²³ are examples of portals, while AOL is a kind of portal as well-- a customised interface to the Internet designed to make the user feels comfortable and to make it easy to find things.

The main disadvantage by using portals consists in the content dictation that the user is able to find. They are a good aid for new users to get acquainted with what is out there and what can be done, but at the same time, they only serve them a slice of the Internet. Portal sites often point the user to sites that live under the same corporate organisation as they do, rather than to the most relevant or informative site on a given topic.

6.1 Oracle Portal-to-Go

Portal-to-go, developed by Oracle²⁴, is a server technology for the use of mobile devices such as Palm, WinCE and wireless phones connected to the Internet. Oracle has developed the prototype of Portal-to-Go together with the Sweden's mobile operator Telia Mobile and actually is available a Beta version of the product. Portal-to-Go was formerly know by the Oracle codename: "Panama". Figure 5 depicts the architecture of the Oracle development platform.

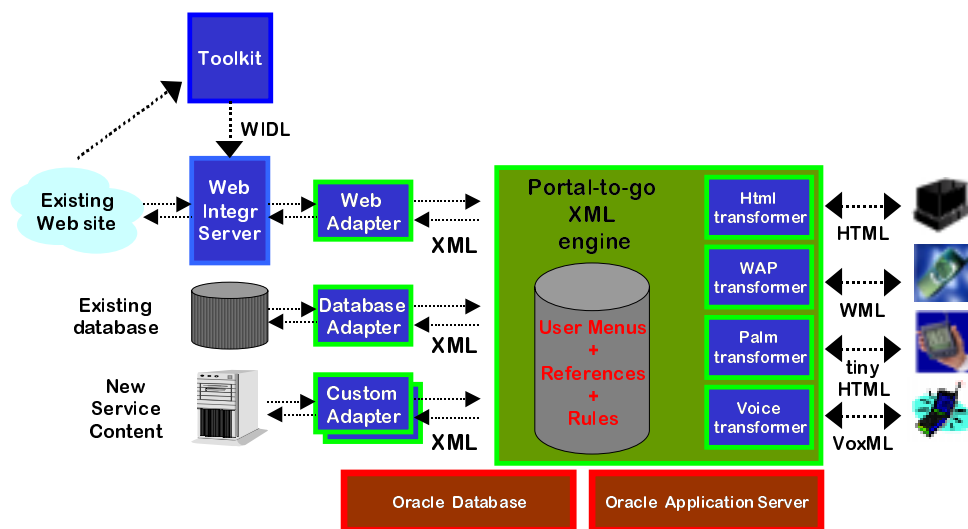


Figure 5 - Oracle development platform

Portal-to-go consists of a service creation environment and a Portal functionality.

²² (<http://www.yahoo.com>)

²³ (<http://www.excite.com>)

²⁴ www.oracle.com

Portal-to-go service creation tools dynamically transform existing Internet content to any native format for a specific mobile device, so Web services that previously only could be accessed from PCs are accessible from different devices. The Internet content is transformed in a generic XML format, and then any device specific output desired is generated. Examples of markup languages supported are HTML, WML, HDML, TTML, Pager-text and VoxML. This approach of a generic Web service description in XML also makes the service available in any markup language spanning from WML for mobile phones over HTML for browsers to VoxML for IVR (Interactive Voice Response) systems. The creation of a service is a very simple task that includes analysing the Web site to be used. The analysis can be done either by a system administrator for more complex transaction based Web services or by the end user for more simple types of information services.

Portal-to-go Portal functionality allows the end users to personalise the start page and the services that they can access from their mobile device(s). By using a browser the users can log on to the portal and add their own favourite links to be used from the mobile device. They can organise the services or the links the way they prefer, effectively creating their own personalised menu structure. The user can personalise individual services by deciding whether information should be entered from the mobile device or stored securely on the portal as part of a personalised Internet service.

The development platform includes also interfaces to non-Internet related information such as any database data and any XML based data. A set of XML based API allows any one to create mobile applications. Since Portal-to-Go has knowledge of the different markup languages and devices for the mobile market content developers the developers don't need to know anything about WML, HTML, HDML, TTML, Pager-text and VoxML and the related devices. When new phones and markup languages are developed its just a matter of downloading the new device transformer and load it into Portal-to-Go to support the new device without rewriting or changing anything in the existing content or portals.

In addition to the described request/response model Portal-to-Go has an asynchronous push mechanism to allow push of any Portal-to-Go service to the mobile device. Portal-to-Go includes push gateways to SMS and email. Since Portal-to-Go handles all data in XML event based push can easy be configured from the end user portal by setting conditions on the data returned from a service before it is pushed to the user.

Oracle Portal-to-Go is implemented using a scalable architecture using Oracle 8i, Java and XML which allows it to be deployed at small corporations wanting to create their own mobile content as well as mobile operators.

6.2 TelePortal

TelePortal is a software solution developed by Lucent²⁵ to expand Web access to customers who don't own computers or mobile phone users who may be away from their computers. After retrieving appropriately formatted Web information TelePortal uses speech technology to deliver it to users via a fixed or mobile phone. Figure 6 describes TelePortal solution.

²⁵ www.lucent.com

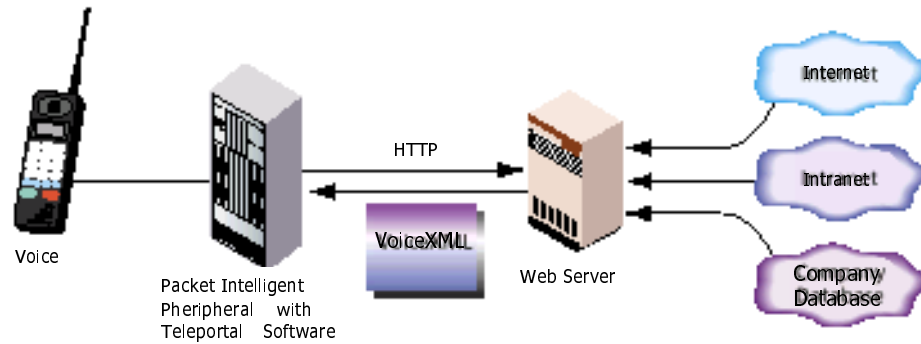


Figure 6: TelePortal situation

The TelePortal software translates Voice eXtensible Markup Language (VoiceXML) commands and uses Lucent speech technology to deliver the Web page content via the phone. TelePortal solution uses Lucent Technologies' for speech recognition technology that lets users speak naturally and in phrases, rather than responding to voice prompts. Since the Web pages and the TelePortal solution delivery mechanism are separate, both content and telecommunication providers can concentrate on developing what they know best. The telecommunication services consider partnering with Web content providers to offer their customers Web access via the phone using their delivery mechanism. The Web contents providers consider offering their customers the convenience of accessing information via phone; they can change the content when necessary, without impacting the delivery system.

TelePortal solution runs on Lucent Technologies Packet Intelligent Peripheral platform that can support multiple services, alternatively the telecommunication and the Web contents providers you can migrate the current Lucent Compact SN/IP to the Packet IP. The Compact SN can support Intelligent Network services under the control of service logic residing on the Lucent Service Control Point, Lucent database server.