WEB ENABLED TELECOMMUNICATION SERVICE CONTROL USING VOXML

Bilel Guedhami, Cornel Klein, Wolfgang Kellerer
Siemens Information & Communication Networks,
Hofmannstr. 51, 81379 Munich, Germany
and
Munich University of Technology, Institute of Communication Networks
Arcisstr. 21, 80333 Munich, Germany

Email: Bilel.Guedhami@icn.siemens.de, Cornel.Klein@icn.siemens.de and Wolfgang.Kellerer@e-technik.tu-muenchen.de

Abstract: Techniques for enabling voice access to the World Wide Web have been extensively investigated both within the research community and industry. The voice markup language VoxML[1] has been proposed as a markup language enabling the HTML-like description of voice services by describing such services as "dialogues". By means of some sample services, we show that VoxML would highly benefit from additional language features for call control. We propose a set of tags that allow to access call control from within VoxML dialogues, and describe a prototype system that implements these tags.

Keywords: VoxML, Service Architecture, Web, Call Control

1. INTRODUCTION

Almost every day one can read a press article reporting about someone who got rich within some months simply by opening up a new ".com"-service, and almost every day several dozens of new such services are

[1] VoxML is a trademark of Motorola Inc.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35522-1_37

H. R. van As (ed.), *Telecommunication Network Intelligence*

deployed on the world wide web. Besides the commercial attractiveness of offering WWW services, from a technical point of view, the rapid pace of growth of the WWW is spurred by the fact that web programming is relatively simple: only a little know-how in HTML authoring and database programming are needed to develop a new internet service. For this reason, web-based services can be deployed by almost everyone, simply by registering a domain address and connecting a new server to the internet.

Compare this with approaches for deploying value-added services within the public switched telephone network (PSTN), such as the IN (intelligent network) architecture. Although offering clear advantages w.r.t. reliability, security and access to billing facilities, the required infrastructure is relatively expensive, controlled by few telecom operators, and the service development requires programming proprietary APIs. For this reason, an approach combining the best of both worlds would be highly desirable.

Access to the world wide web is, however, still limited mainly to PC-based clients, due to the nature of HTML and the web-pages expressed therein, which both assume that large, graphical displays and sufficient processing power on the clients are available. Several attempts have been made to overcome this deficiency. Most famous is WAP, an architecture and a suite of protocols which are tailored for bringing the web to mobile phones, which have both limited display capabilities as well as limited processing power. Less known, but interesting because of the immense amount of installed clients, are approaches that aim to provide voice-based internet access from ordinary telephone sets.

The objective of so-called "Voice Markup Languages" such as VoxML [VoxML] is to provide, similar to HTML, a simple approach for developing speech applications and making them accessible to a broad audience via any voice-enabled client (phones, mobiles). The idea behind voice markup languages is that a "voice surfer" dials into a "voice browser" via the public switched telephone network (PSTN). The voice browser accesses, via HTTP, web servers in the public internet or within corporate intranets. The content, which is fetched from these web servers, which is expressed in VoxML and which is structured into so-called "dialogues", is interpreted and transformed by the voice browser into speech signals. For output, speech synthesis or the play back of prerecorded audio-files is used. For input, technologies for speech recognition or for DTMF-decoding are used. VoxML does perfectly fit for information services such as

- voice-surfing the web (via dedicated VoxML pages or via HTML to VoxML gateways),
- inquiries of email (via VoxML gateways to email systems or via VoxML interfaces of existing webmail systems),

- access to information systems e.g. for travel information, weather information, stock quotes etc.

However, services which require access to network features such as call-control or billing can not be implemented conveniently this way, i.e. only by accessing platform-specific APIs. This makes VoxML inappropriate for services like

- call centers, which require redirecting calls from voice-surfers to other telephone lines or
- services which require connecting voice-surfers with other subscribers, like e.g.
    - establish a call to a subscriber after this subscriber has been found by a VoxML-based lookup of a telephone directory
    - establish a call to the originator of a voice mail after a VoxML based inquiry of the voice mail box.

In this paper, we will propose an extension of VoxML with features for call control. This will allow the portable and platform-independent description of the above-mentioned kinds of voice-based services. By integrating call control into VoxML, content authors – with little know-how in programming – will be enabled to develop web-based speech applications by separating the underlying call control from content. In particular, these VoxML based service descriptions will be independent of any underlying network APIs (TAPI, JTAPI, Parlay), signaling protocols (H.323, ISDN signaling) or service control platforms (e.g. SCP, CTI Servers).

The paper is structured as follows: In Section 2, we will review existing approaches for voice-based access to web-based information systems and discuss some services to show the need for an enhancement of VoxML. In Section 3 we will propose a set of language extensions ("tags") for VoxML. In Section 4 we will present the architecture of an enhanced VoxML enabled telecommunication system and describe our prototype implementation. In Section 5 we will discuss the benefits of our approach, and in Section 6 we will discuss related work. We will conclude in Section 7 with a summary and an outlook on future work.

2. VOICE-BROWSER BASED ACCESS TO WWW BASED INFORMATION SYSTEMS

The main idea of voice browsing is that a "voice surfer" dials into a "voice browser" via the public switched telephone network (PSTN) (Fig. 1). The voice browser accesses, via HTTP, web servers in the public internet or within corporate intranets. However, the proposed solutions differ in the

form the content is stored on the web server – HTML or specialized markup languages like VoxML – and the intelligence which is built into the browsers.

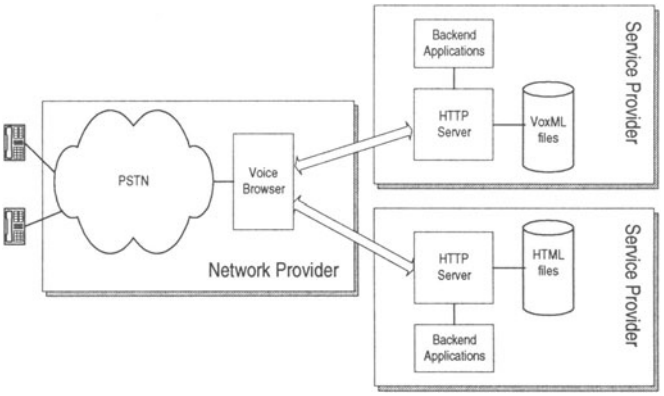

Figure 1. Voice Browsing Service Architecture

Voice browsers are browsers that allow people to access the web through voice interfaces. A combination of keyboard and speech recognition is usually used for input, and speech synthesis and pre-recorded sound are used for output. An extended concept of a voice-browser may also control small displays on telephone when these are available. Despite these limited capabilities for input- and output, voice browsers represent a new and attractive alternative to the classical browsers (that use screen and/or keypad). They make the need for the hands and the eyes unnecessary and reduce the required equipment for web browsing to a telephone.

Beside opening the web doors for people with telephone access, voice browsers also provide a field for new ideas for web-based applications and services. Call centers and hands-free mobile applications especially in automobiles can make use of this speech technology.

Today, almost all the information published in the web is authored in HTML. So it is attractive to offer voice access to HTML content. As HTML is designed with a visual interface in mind, it is difficult to create high-quality voice services using today's HTML standards. Some people suggest the extension and the adaptation of HTML to the new requirements. Others propose to create a new mark-up language that target the voice medium precisely and directly.

2.1 Extension of HTML

HTML (HyperText Mark-up Language) is a set of "mark-up" symbols or codes inserted in a file intended for display on a World Wide Web visual browser. The mark-up specifies for the browser how to present a document on a screen, including words, images etc.

HTML has been developed with visual rendering in mind. As a result, many of its elements would not be amenable to speech rendering (e.g. image maps). Other elements could be misinterpreted or even senseless for voice browsers (e.g. frames). In addition, lots of voice specific aspects cannot be supported since there are no corresponding HTML tags. For example, using a specific recognizer or synthesizer that associate specific recognition grammar with input elements and with the control over the synthesizer volume, speed, pitch, etc., and also the interaction for timeouts, errors etc.

Obviously, HTML in its current form will not be of great help when authoring web sites for voice applications. As they assume it to be unlikely that web designers will develop different pages for both visual and voice browsers, some researchers suggest the extension and reform of HTML to support voice browsing. These "pro-HTML" researchers underline that adding voice capabilities to the web should be regarded as a natural evolution of the web and not as an industry specific derivative, as this could be the case when resorting to a new mark-up language. They assume that a combination of the following techniques would provide a suitable solution for the voice extension of the web:

- *Extension of HTML:* Extensions of HTML such as Aural Style Sheets, e.g. [Fer98] (which are part of the Cascading Style Sheets, Level 2 specification [RHJ99, CSS]) provide a level of control in spoken text. With the use of an aural style sheet, authors are allowed to specify characteristics of the spoken text such as volume, pitch, speed and stress. It is also possible to indicate pauses, insert sound files (analogous to icons in HTML) and show how certain phrases, acronyms, punctuation, and numbers should be voiced.
- *Style guides for HTML authors [Gun98]:* Styleguides provide rules for writing HTML-files which can easily be translated into useful dialogues by voice browsers. An example is to provide useful text choices for links, where text descriptions instead of URL's make the links easier to speak. Also, names of persons are usually simpler than their e-mail-addresses. Furthermore, ambiguity could be reduced by avoiding same words for different links. Here are some examples:

This version: http://www.w3.org/WD-USERAGENT-19980814
Instead of:
This version : http://www.w3.org/WD-USERAGENT-19980814

Select local news or national news.
Instead of:
Click here for local news, or here for national news.

- *Intelligence in Voice Browsers [Gun98]:* When browsing into Web pages, it is easier to determine their structure and retrieve the relevant information by visually scanning them rather than by listening to all their content read by voice browsers. This could turn to be very uncomfortable especially with long pages. Voice browsers can apply intelligent abstraction techniques such as forming a structure based on the <H1>, <H2>, ... headers or listing or the links on a page, to provide a kind of a structured summary to the user. This suggests that the author uses the HTML elements properly and meaningfully. For example he should make useful choices for link text or makes use of the SUMMARY attributes for tables.

The main point is whether these proposed strategies together with appropriate HTML extensions would be sufficient to support voice access to the web, or whether new specialized formats need to be created. The question that has to be asked is whether people really want general telephone browsing, or do they just want access to particular important information in more than one way.

2.2 Voice Markup languages

Based on the fundamental requirements for a mark-up language that supports voice access to the web, we have found that the extension of HTML has some important limitations. HTML could not be easily extended in ways that would make voice browsing possible. In addition, generating a dialog system from a graphical representation can easily fail to create a usable or useful system. We also suppose that the majority of people wanting to access the web via voice interfaces are not interested to hear the content of some people's private sites "spoken" on the telephone but rather aim to make use of some important and helpful documents and services on the web. Such documents and services will have to be designed with a specific mark-up language for the interactive voice rendering. The ideal would be to develop a mechanism that allows both graphical web pages and telephone dialog systems.

Experience shows that the complexity of dialogs and interactions can overwhelm a programmer not skilled in both concurrent systems and human-

computer interfaces. A structured mark-up language is a convenient way to hide hardware complexities, to abstract away the difficulties of concurrent programming and to codify proven principles of IVR (Interactive Voice Response) design. The mark-up language should allow the designer to concentrate on the essential matter: the contents, choice of pre-defined dialogs and associated help menus and prompts.

For this reason, several companies have proposed voice markup languages. In this paper, we will stick to VoxML [VoxML], which has been developed by Motorola and which is used within our prototype. A comparison with the upcoming industry standard VXML is given in Section 6.

It is important to note that several concepts and elements of HTML apply in the voice world. For example the LINK element in the header section that define the relationship to other documents, the FORM element that defines the name/value pairs to be returned to the server. The incorporation of HTML elements in the new mark-up language and the adoption of some of its basic concepts makes it easier to learn and to use.

Voice Markup languages will enable a greater flexibility in accessing internet resources, as everyone can launch a variety of information and communications applications from anywhere provided that one could access a telephone. They also offer new business opportunities for content developers and network operators and they allow rapid and easy application development for voice access. However, the wide spread of voice browsers would depend on the ease with which web designers can add speech support to their site.

2.3 Service Examples

VoxML enables a large spectrum of voice services to be created. The main characteristic of these services is that they provide an interactive dialog system for users, known also as IVR (Interactive Voice Response). Users would then have access to multiple sources of information that can be adapted to their personal needs. This adaptation occurs dynamically within the user-browser conversation, as the user can decide which information is relevant to him and he wants to hear.

Since the access device for the voice services is the ordinary phone, service providers can extend their function as an information source to provide telephony and networking applications, such as setting up and forwarding calls, arranging and managing voice or video conferences (depending on the user calling device) and sending mails.

What is currently missing in VoxML are language constructs for call control. The following two examples services indicate that an appropriate extension of VoxML would be highly beneficial:

Information Inquiry and Call Management Service: A company may offer a voice-based inquiry service for telephone and e-mail addresses of its employees. By dialing a special service number, users can look up the corporate's directories (which is e.g. stored on an LDAP server) by naming the person through voice recognition or by typing it on the phone keypad. Users can then ask the system to establish a connection to the employee.

Note: VB stands for Voice Browser. U means User.

VB:	*Welcome to Siemens. You can use our service to get the co-ordinates of our employees such as their phone, fax or email addresses. We can as well transfer your call to any destination or set up conferences with whom you wish. Please say "search", transfer" or "conference".*
U:	*search*
VB:	*Please say the surname of the person you wish to look for, or type it on the keypad of your phone*
U:	*Mayer*
VB:	*Please say the first name of the person you wish to look for, or type it on the keypad of your phone*
U:	*Peter*
VB:	*Are you looking for the phone, the fax number or the email address?*
U:	*phone number*
VB:	*The telephone number of Peter Mayer is 722-12345*
	If you wish to get connected to him now, please say connect. And if you want to have him as a part of conference please say conference.
U:	*connect*
VB:	*I'll try to establish the connection now. Please don't hang up...*
(Trying to call Peter Meyer)	
VB:	*I am sorry, the line is busy. If you wish to leave him a message please say message. If you wish to send him an email, please say email*
U:	*message*
VB:	*You can speak after the sound signal for a maximum of 30 seconds*
U:	.....
(After longer breaks or 30 seconds)	
VB:	*Thanks a lot for exploiting our service. Can we help you any further?*
U:	*No.*
VB:	*Bye.*

Figure 2. Information inquiry and call management service

```
<?xml version="1.0?>

<DIALOG>
 <STEP NAME="init">
  <PROMPT> Welcome to Siemens. You can use our service to get the co-ordinates of
  our employees such as their phone, fax or email addresses. We can as well transfer
  your call to any destination or set up conferences with whom you wish.
  </PROMPT>

  <INPUT TYPE="NONE" NEXT="#main_menu"/>
 </STEP>

 <STEP NAME="main_menu">
  <PROMPT>  Please say search <BREAK/> transfer <BREAK/> or conference
  </PROMPT>

  <INPUT TYPE="hidden" NAME="inputn_first_name" VALUE="FIRST_NAME"/>
  <INPUT TYPE="hidden" NAME="inputn_last_name" VALUE="LAST_NAME"/>
  <INPUT TYPE="hidden" NAME="search_type" VALUE="NEW"/>

  <INPUT TYPE="optionlist" NAME="service">
   <OPTION NEXT="#search_service"> search </OPTION>
   <OPTION NEXT="#transfer_service"> transfer </OPTION>
   <OPTION NEXT="#conference_service"> conference </OPTION>
  </INPUT>
 </STEP>

 <STEP NAME="search_service">
  <PROMPT> Are you looking for the phone, the fax number or the email address?
  </PROMPT>

  <INPUT TYPE="optionlist" NAME="search_option">
   <OPTION NEXT="#firstname_request" VALUE="phone number">  phone </OPTION>
   <OPTION NEXT="#firstname_request" VALUE="fax number">    fax   </OPTION>
   <OPTION NEXT="#firstname_request" VALUE="email address"> email </OPTION>
  </INPUT>
 </STEP>
....
```

Figure 3. Sample VoxML Script

The convenience and quality of this service can of course be extended in many ways. For instance, to improve the recognition, the server can compare the recognized name to the entries of its database. In case of slight differences the server can correct some input faults. Moreover, if it is not possible to setup the connection, users are offered to send an email or voice mail alternatively.

Figure 3 shows a sample VoxML script, and Figure 2 the corresponding interaction between a user U and a voice browser VB of such a service.

Virtual Call Center: VoxML can provide an alternative to realize small virtual call centers with simple functionality. A virtual call center is a call center that is composed of multiple units that have not to be located geographically at the same place. These tiny units can represent call centers on their own or unique persons working for instance at home. Callers have to dial just one service number. A server would then distribute the incoming calls over the several associated units. Virtual call centers represent a considerable way to achieve great flexibility, increase productivity, and reduce costs when creating a call center.

3. EXTENSION OF VOXML WITH FEATURES FOR CALL CONTROL

As we have seen, VoxML in its current form enables already a large spectrum of voice services to be created, in particular access to web-based information systems where users have access to multiple sources of information that can be adapted to their personal needs. This adaptation takes place dynamically within the user-browser conversation, as the user can decide which information is relevant to him and he wants to hear.

However, to define more sophisticated telecommunication services like conference calls within a voice browsing session, VoxML has to be extended with call control commands. By analyzing several services, we have identified the following new tags that can form the basis for a future standardized extension of VoxML. In particular, we propose the following tags:

- TRANSFER: Transfer a call to another destination
- MCALLS: Transfer a call to some destination within a group of destinations (e.g. of call center agents)
- CONFERENCE: Set up a conference between more than two parties
- SEARCH : Search a data base
- EMAIL: Send an email.

In the sequel, we will describe these tags, which we have included in our prototype voice browsers system, in more detail. Since all functions are selected by user input the tags are defined as part of the INPUT tag family.

3.1 TRANSFER Tag

Syntax:

```
<INPUT   TYPE="TRANSFER"     NAME="name"
         DEST="phone number">
```

Attributes:

DEST: The telephone number the call is transferred to.

NAME: The name of a VoxML global variable, to be set when accomplishing the transfer action. Its value can be retrieved within further steps.

3.2 CONFERENCE Tag

Syntax:

```
<INPUT   TYPE="CONFERENCE"   NAME="name"
         PARTIES="party, party,..."
         CONFTYPE="type">
```

Attributes:

PARTIES: List of the conference participants

CONFTYPE: The type of the conference, e.g. "voice" or "video"

The CONFERENCE tag is quite different from the TRANSFER tag since here we aim setting up one or more completely new connections in addition to or beneath the existing voice browser session.

3.3 EMAIL Tag

Syntax:

```
<INPUT   TYPE="EMAIL"       NAME="name"
         FROM="name@host"
         TO="name@host"
         SUBJECT="subject"
         ACK="value"
         ATTACH="file.doc"
         TEXT="body">
```

Attributes:

FROM: email address of the sender (user)

TO: email address of the recipient

SUBJECT: subject of the email

ACK indicates if an acknowledgement message is to be sent to the user account

ATTACH: list of files to be attached
TEXT: The body of the email

The EMAIL tag goes beyond the purpose of providing telecommunication services. We have included this tag on the one hand to show the flexibility of the enhanced VoxML based telecommunication service control. On the other hand email services are an essential improvement for today's telecommunication services. Just have a look on the success of the SMS service in GSM.

3.4 MCALLS Tag

In order to create the service "virtual call center", it is necessary to be able to connect a caller to the next available call center agent. This requires that the voice browser should be able to try to simultaneously establish a connection with any agent within a specified group. This can be described using a new tag <MCALLS>. "mcalls" means "multiple calls". This is similar to the "fork" command within the IETF Session Initiation Protocol (SIP), which deals with the same scenario [SR99].

The MCALLS tag has got the following syntax:

```
<INPUT   TYPE="MCALLS"      NAME="name"
         PARTIES="number, number,..."
         SIMULT="4"
         LOOP="yes"
         TIMEOUT="time">
```

Attributes

PARTIES: The list of telephone numbers of the concerned units
SIMULT: Indicates how many numbers the browser tries to reach simultaneously.
LOOP: Can take the value "yes" or "no". "yes" means the voice browser should go through the telephone numbers list once again, if he fails to setup a connection during the first attempt.
TIMEOUT: Indicates the time interval, after which the connection setup procedure should be aborted.

3.5 SEARCH Tag

Database access represents today one of the main features of the services offered on the web. Search machines, online dictionaries, booking services and innumerable other applications live on the dynamic information resources provided by databases.

Voice-based web services and screen-based web services should be all the same except for their access interfaces. In this way it is obvious to integrate a

database functionality also in the voice based system. To support the developers with an easy way of handling data base requests we integrated a basic database functionality on the mark-up language level in contrast to HTML-based services, which use CGI-scripts whenever database queries have to be sent. Precisely, a click on an appropriate button (often labeled "search", "start" or "send"...) in the service page, launches a CGI-script, that collects and analyses the information filled in the entry boxes by the user, conducts the communication procedure with the database to finally generate new HTML pages containing the new acquired data.
Our approach is to enhance VoxML with a new tag, which provides to the service designer the following basic database functions:

- search a given database for a specific key value
- retrieve the matching entries successively
- refine search results

Based on these functions we have defined a new VoxML tag:

```
<INPUT TYPE="SEARCH"        NAME="name"
       DB="database"  QUERYNAME="search1"
       INPUTNAME="name"
       INPUTVALUE="value"
       OUTPUTNAME="name"
       OUTPUTVALUE="value"
       SEARCHTYPE="type">
```

Attributes:

NAME: The name of a global variable, which is set according to the result of a data base search, e.g. no_entries, one_entry.
DB: Name of the database the request goes to
QUERYNAME: Label of the search request for identification
INPUTNAME: List of the field names to be checked
INPUTVALUE: List of the field values
OUTPUTNAME: List of the searched field names
OUTPUTVALUE: List of the variable names where the searched values are to be stored
SEARCHTYPE: Type of the current search (NEW, REFINE,...)

In order to create a flexible voice system, the mentioned database functions are to be provided by an independent module, called the directory server. The communication between the VoxML browser and the directory server is message based. Thus, the role of the browser is limited to parsing the VoxML files, generating and sending the search messages to the directory server, and interpreting the returned messages. This logical architecture also implies a possible physical separation, as the two modules could be located on different servers (see Figure 4).

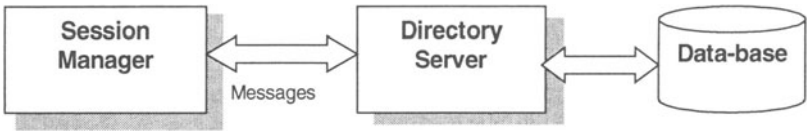

Figure 4. Principle Architecture of the database query enhancement

4. ARCHITECTURE OF A VOXML ENHANCED TELECOMMUNICATION SERVICE NODE

In this chapter, we present the architectural concept of a voice browsing platform, enabling the creation of voice-based telecommunications services. Furthermore we describe a prototype implementation that has been built according to the general concepts.

The platform is composed of several basic units (see Fig. 5). Their interaction and data flow is controlled and supervised by the central unit. The platform also includes software and hardware components for data (voice and tones) input and output such as Telephone API (TAPI) compliant hardware or speech synthesis and recognition tools.

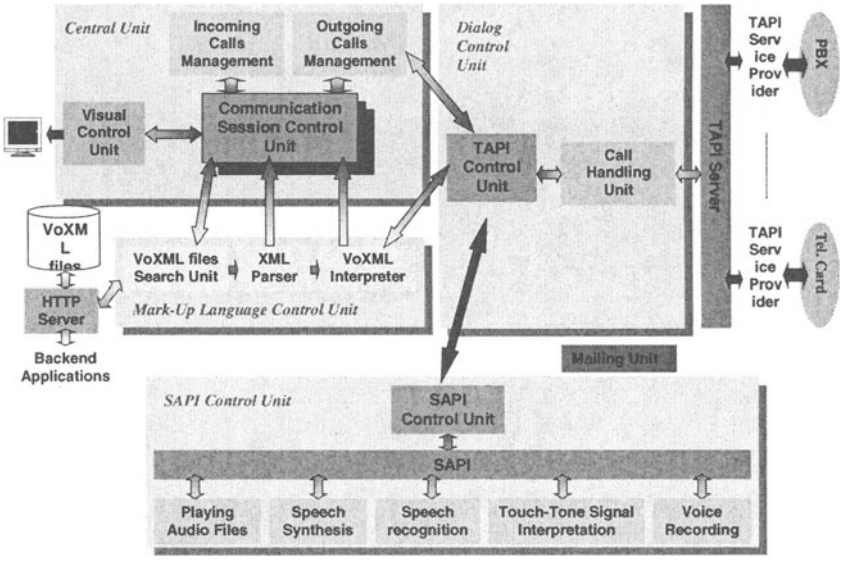

Figure 5. Architecture of an enhanced telecommunication service node

4.1 Microsoft TAPI

The Microsoft Telecommunication Application Interface (TAPI) is an open industry standard for the support of telephony service control. It provides network and device independence and supports a great variety of telephone features.

The MS TAPI model supports four different physical configuration alternatives. The first is phone based. That means the telephone is in between the application running on a PC and the switching system. The second alternative is the PC based configuration, in which the PC is the means to access the telephony lines. In the shared or unified line alternative the PC and the phone have the same access to the phone lines.

All of the above configuration types only allow the control of one telephone line per workstation. The last of the four, the multiline configuration, supports the control of many lines and is therefore suitable for the realization of PBX applications or for the discussed voice server.

For the programming of the controlled telephony resources the MS TAPI provides a number of functions, which support the development of services on telephone networks. For simple basic telephony services the TAPI basic functions are sufficient for the control of incoming or outgoing calls. Advanced services like call forwarding, suspend or conference control use the supplemental telephony functions. The class of extended telephony services is for device specific functions of hardware vendors.

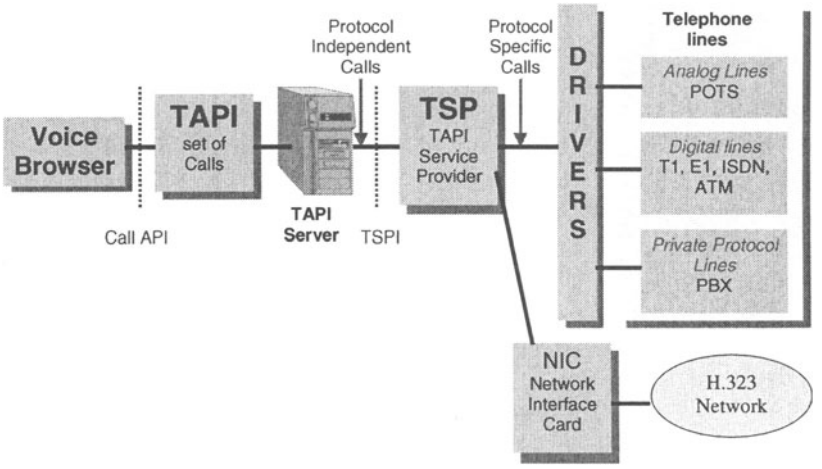

Figure 6. MS TAPI Architecture

The control of a telephony resource e.g. a telephone line or a PBX is illustrated in Fig. 6. An application, which may be a voice browser for example, uses the TAPI functions. The TAPI server transmit these function calls to the TAPI service provider (TSP). The TSP translates protocol-independent functions in specific functions of the underlying driver that controls the telephone resource. A telephone resource may be a simple analogous line or a digital PBX or even a H.323 network. MS TAPI is supported by a large number of resource providers. Most of the PBX products on the market provide a TAPI interface.

4.2 Components of the Telecommunication Service Node

Coming back to our voice browsing platform from Fig. 5, whenever a call request is detected by the TAPI control Unit, the central Unit starts a new session. It is then the role of the Mark-up Language Control Unit to get the VoxML files and their DTD (Document Type Definition) from the local memory or from a remote voice-service server via an HTTP-server. The browser has then to interpret the parsed VoxML resources and to deploy the TAPI and SAPI (Speech API) units that provide the call control and speech control mechanisms. The directory server assures database access functionality. The mailing unit helps to realize voice mail services. More precisely, the different units serve the following tasks:

Central Unit. Its main function is to manage the user sessions and to control the underlying units. This includes launching their mechanisms, checking their state of function and coordinating the communication between them.

Mark-up Language Control Unit. This Unit consists of three main modules: the VoxML-Files Search Module, the XML Parser and the VoxML Interpreter.

The VoxML-Files Search Module is responsible for seeking after VoxML files and their DTD. This may imply setting up connections to remote HTTP-servers. The VoxML files are retrieved when a user session is launched or during the interaction itself, if the browser steps on a link request. The VoxML files are to be handed over to the XML parser.

The XML parser checks the received VoxML source code for validity and conformance with the DTD. If the document is well-formed, it is translated into a parse tree, handed over to the VoxML interpreter and an indication is sent to the central unit. Otherwise, the central unit is informed of the parsing failure. The error handling procedure is then to be applied.

The VoxML Interpreter is the most important element in the voice browser. It processes the received DOM tree from the XML parser and starts the related applications. It holds the control on the SAPI and TAPI control units as well as the directory server and the mail services unit, by sending them commands and interpreting their responses and indications. It analyses user inputs delivered by the SAPI Control Unit and generates output commands.

TAPI Control Unit. The TAPI control unit is the intermediate unit between the VoxML Interpreter and the TAPI compliant hardware. Its tasks involve the execution of the network access commands received from the VoxML Interpreter by translating them to TAPI calls and sending back confirmation messages, state and failure indications. The TAPI control unit has to interact with the underlying TAPI hardware and handle all the events reported by the TAPI call-back mechanism. For instance, in case an incoming call request is detected, the central unit is informed, so that a new session can be started.

SAPI Control Unit. The SAPI control unit has the direct access to SAPI. So it transforms the received commands from the VoxML Interpreter into SAPI specific commands, that control the various speech technology engines. It detects user inputs and forwards them to the VoxML interpreter.

The main tasks of this unit include the speech synthesis, playing audio files on an established connection, voice recognition and recording.

Directory Server. The directory server performs database access and realizes basic functions, such as searching for specific field values in database records or retrieving and refining results. It locally stores the results of search queries, to use them for refinement purposes until it is overwritten by a new query or the session ends up.

Mail services Unit. This unit is contacted each time an email is to be sent.

Visual Control Unit. This unit represents a monitoring system for the voice browser and its running applications. It may also be used for debugging purposes. It shows the working units, the progress of the calling procedures and displays the received messages from TAPI and SAPI interfaces. It provides also a graphical overview on the actual proceeding connections, the number of the available and active (or inactive) line and phone devices, and the incoming and outgoing calls.

4.3 Realization

To show the feasibility of our approach we have realized the concepts described above in a prototype implementation. To support rapid prototyping we have based our implementation on existing components. It is two main components we need. First we have to take a system that allows control over telephone calls. This enables the development of services within the telephone network. Second we have to integrate a voice browser system. This system has to be enhanced by the new defined tags and has to be connected to the call control.

4.3.1 Siemens VoxPortal

VoxPortal is a commercially available product that has been developed by Siemens Information and Communication Networks [VoxPortal]. It represents a powerful and flexible IVR system basing its interaction rules on the VoxML language. Consequently, the VoxPortal could be considered as a voice-based access tool to any type of information, provided that it has been previously translated to VoxML.

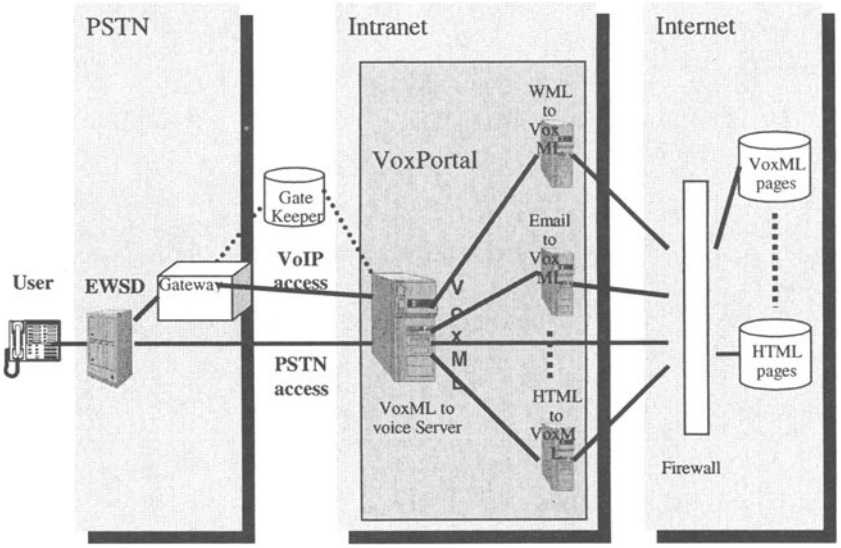

Figure 7. VoxPortal Architecture

The VoxPortal architecture is depicted in Figure 7:

- *PSTN:* The user dials into the VoxPortal via the public switched telephone network. Depending on the operator's infrastructure,

VoxPortal is connected to the PSTN either via E1 lines or via a VoIP Gatekeeper (H.323). To interact with the VoxPortal, the user can make use of DTMF dial tones (currently supported by nearly any phone) or voice commands (planned in future releases).

- *VoxPortal:* The VoxPortal, which runs a VoxML browser, represents a bridge between the voice world and the internet world (HTML, VoxML)
- *Add-on servers:* Several application-specific add-on servers enable a great variety of interactive voice services to be realized such as remote email access or simple retrieval of HTML pages.

Altogether, VoxPortal represents a perfectly suitable platform for the realization and the demonstration of our concepts and ideas.

4.3.2 Enhancement of the Vox Portal

The commercial implemantion of the existing VoxPortal product has been the basis for our prototype. For the realization of our prototype we had to integrate the telephone network via TAPI to this VoxPortal system. In addition we had to add and to supplement several modules. Fig. 8 shows the main points of work for our demonstration application.

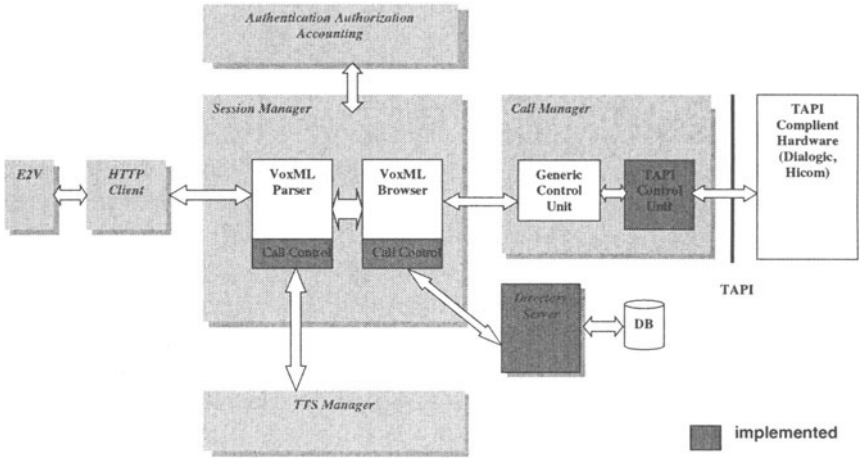

Figure 8. Integration of the VoxPortal and a TAPI based call control

The VoxML parser as well as the VoxML Browser had to be enhanced to include the new call control features. All actions resulting from the new tags had to be included in the session manager system.

For the interaction with the telephone network we have chosen the MS TAPI since this is supported by our switching hardware consisting of a dialogic board. A new TAPI module had to be developed acting as an application on the TAPI interface to generate and control calls from the voice browser. The TAPI module had to be integrated in the existing Call Manager module in addition to the existing module controlling the functions of the dialogic board for incoming calls.

For the demo system we decided to include the directory server and the data base in the voice browser system. The directory server interacts as a client with a database to handle information queries.

In order to be able to test and to demonstrate the system, we have implemented several applications, in particular the one depcited in Figure 3, as enhanced VoxML scripts.

5. OPPORTUNITIES AND BENEFITS OF USING ENHANCED XML VOICE INTERFACES

Beyond the new features introduced for call control, the presented system concept reveals opportunities that are also applicable in a broader context. The general ideas of our proposed approach can be applied in many other areas of telecommunication service development as well.

5.1 Paradigm Shift

The voice browser concept (as is, or applied to telecommunication service control) changes the traditional Web browser/server architecture. In the traditional HTML based world the information is displayed on a common Web browser like Netscape Communicator or Microsoft Explorer. The web browser is running as an application on the user terminal. The only way a information service provider can influence the user domain is to send cookies or to provide applets for download. As we can see in Fig. 9 this strict separation of domains between browser and server that means user domain and service provider domain is totally changed with the voice browser system.

Since the user terminal is no longer an intelligent end system but an ordinary telephone system, mechanisms in the network provider domain have to be provided to realize voice browsing. That means in other words that the voice browser is located in the service provider domain. This offers new opportunities for the service provider who offers a voice WWW portal. Examples are:

- New browser updates e.g. for new versions of VoxML can be introduced more rapidly, because only the browsers of the service providers have to be updated.
- This way, new tags for the creation of new services can be introduced more rapidly.
- The browser is located in a trusted area, enabling an easier introduction of securitiy-critical features such as billing (difficult in the internet).
- Furthermore, user can be bound to voice WWW portals by offering personalized services e.g. personal bookmarks access from everywhere.

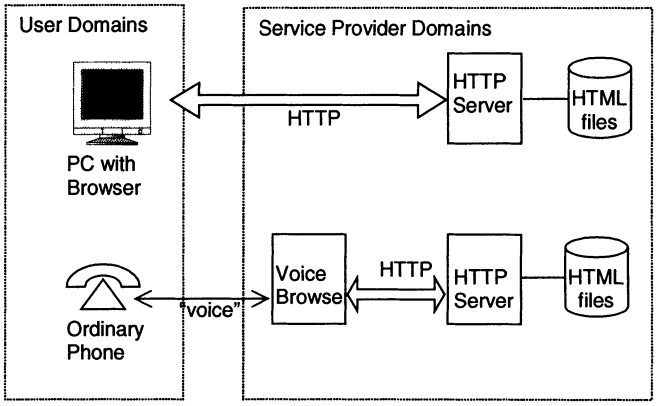

Figure 9. Voice browsing changes browser/server responsibility

When we look back on Fig. 1 we can see, that the voice portal service provider does not need to be the same as the content service provider (VoxML content) but also a network provider can earn revenue by this new role.

5.2 XML based service definition

As we have shown, the XML based VoxML language allows a very flexible definition of voice accessed information services. We have enhanced the VoxML by new tags to support some telecommunication services, additionally. Looking beyond voice based services the Extensible Markup language XML provides a basis for the definition of telecommunication and information services in general. For example the definition of personalized services like unified messaging in the IETF IP Telephony approach uses the Call Processing Language CPL [RLS99], which is based on XML.

In comparison with traditional service definition methods like the use of SIBs in the Intelligent Network the XML based service definition by scripting languages has a number of advantages:

- XML very well supports representation of structured data (e.g. service logic trees)
- XML allows definition of all necessary tags and is open for all kinds of extensions
- XML is easily readable also for human users
- XML allows easy verification against a Document Type Definition
- XML is fault tolerant by skipping unknown sequences by the parser

6. RELATED WORK

We have chosen VoxML as voice markup language and TAPI for call control as a basis. An alternative would have been to use the upcoming industry standard VXML and/or the PARLAY API.

6.1 VXML

Many companies recognized the importance of accessing the resources of the web by voice interfaces and developed new mark-up languages based on XML to enable easy creation of voice-enabled applications. Recently, AT&T, Lucent Technologies and Motorola decided to form the Voice eXtensible Mark-up Language Forum (VXML Forum) [VXML] to unify their efforts and contribute their technologies and experiences in this domain to the development and promotion of the open standard VXML specification.

AT&T, Lucent and Motorola worked long time on their own independent projects. While AT&T was building a mature phone mark-up language and launching applications, Lucent was continuing on a similar project known as TelePortal. Motorola's language, VoxML [VoxML], has focused on hands-free access for its mobile phone customers and uses speech recognition rather than touch-tones as an input device. With the VXML specification, the three companies hope to leverage the best of their approaches for the benefit of the entire industry. Since its launch in March 1999, the VXML Forum has more than tripled in size by adding 44 leading technology industry players to as supporters including: Conversa, Ericsson, France telecom, Siemens and Sun Microsystems.

Unlike VoxML, VXML contains a <TRANSFER> tag that can be used to forward calls to other subscribers. Not only is this transfer tag reserved for operator calls only, but also more sophisticated tags e.g. for setting up

conferences are still not available in VXML. Moreover, in contrast to VoxML, VXML capable voice browsers are still not available. This has been the motivation for using VoxML instead for VXML as a basis for our work.

6.2 PARLAY

The Parlay working group [PARLAY] was formed in April 1998 by BT, Ulticom, Microsoft, Nortel Networks and Siemens. It aims the development of an API specification that enables applications to control a range of network capabilities and to access network information independently of network specific details. In December 1998, the first version (Version 1) of the specification was released together with an application for demonstration purposes.

The PARLAY architecture is similar to the TAPI architecture, but is targeted towards IN-like service control. Applications developed by a service provider make use of the PARLAY functions that are provided by the network provider on top of his network resources and are converted via resource interfaces in the network specific functions (see Fig. 8). The main difference compared to TAPI is the mechanism how the functions are provided for the application developer. The PARLAY API is therefore separated into two classes: The framework interface and several service interfaces.

The framework interface contains all necessary support functionality for the access control as well as for security, service discovery and maintenance. After successful authentication the service discovery mechanism supports the request of the application for a suitable service interface. The security is an essential feature to imply the adoption of the Parlay API by both service and network providers.

Service interfaces provide the network capabilities of the underlying resources including call control, messaging and user interaction. The Generic Call Control Service (GCCS) for example is based on a third party model, which allows calls to be instantiated from the network and routed through the network. It supports the same functionality like today's Intelligent Networks.

Due to the availability of the TAPI-based VoxPortal, we have based our call control features on TAPI. However, a PARLAY-based realization may be needed for additional features such as third-party call control or access to billing facilities by new tags.

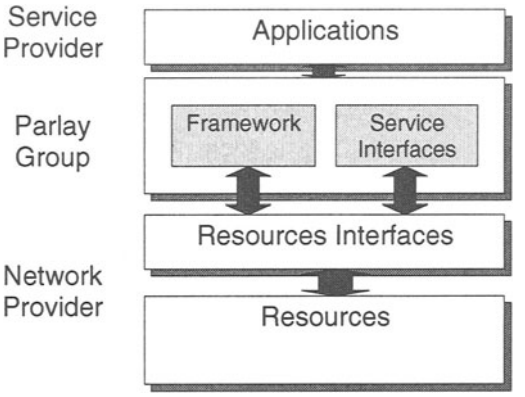

Figure 10. PARLAY Architecture

7. CONCLUSION AND OUTLOOK

We have introduced VoxML and VoxPortal as a web-based language and architecture for deploying voice services to users of voice-enabled clients such as ordinary telephone handsets or mobile phones. By means of some examples, such as call centers, we have seen that VoxML in its current form is not sufficient for the platform-independent description of such services. For this reason, we have proposed some tags as a minimal extension of VoxML which allows to access call control directly from within VoxML. The feasibility of this approach has been shown by extending the VoxPortal from Siemens ICN accordingly. We expect the following benefits from our approach:

- *Platform independency*: New services such as call centers or inquiry systems of telephone dictionaries with automatic call setup can be specified in a platform independent way. In particular, these services can be developed by anyone in a web-like fashion, without consideration of the technology of the underlying voice browsers accessing these services.
- *Separation of concerns*: Bringing new language features for call control into the VXML is a clear separation of concerns, which allows the use of such features by web authors with little or no know-how about programming.

Issues to be investigated further include tags for billing and approaches for personalized services (e.g. via cookies).

8. REFERENCES

[CSS] W3 Consortium. Cascading Style Sheets (CSS). http://www.w3.org/Style/CSS

[Fer99] J. Ferraolo. Scalable Vector Graphics (SVG) 1.0 Specification Appendix E.2 Aural Style Sheets. W3C Working Draft 3 December 1999. Http://www.w3.org/TR/1999/WD-SVG-19991203/access.html

[Gue00] B. Guedhami. Enhancing the VoxML with Language Features for Call Control. Dipl.-Ing. Thesis, Institute of Communication Networks, Munich University of Technology, 2000.

[Gun98] J. Gunderson. WAI Accessibility Guidelines: User Agent. W3C Working Draft, 14 August 1998. http://www.w3c.org/WAI/UA/WD-WAI-USERAGENT-19980814.html

[MSTAPI] Microsoft Windows NT Server TAPI White Paper, Microsoft, 1996.

[PARLAY] PARLAY, http://www.parlay.org, 1999.

[VoxPortal] SURPASS Application VoxPortal. Siemens ICN, product description A30828-X8010-S2-1-7618.

[SR99] H. Schulzrinne, J. Rosenberg. The IETF Internet Telephony Architecture and Protocols. IEEE Network, May/June 1999, pp. 18-23.

[RLS99] J. Rosenberg, J. Lennox, H. Schulzrinne. Programming Internet Telephony Service. IEEE Network, May/June 1999, pp. 42-48.

[RHJ99] HTML 4.01 Specification. W3C Recommendation 24 December 1999. Http://www.w3.org/TR/html4

[VXML] VoiceXML Forum. Voice Extensible Markup Language (VXML). Http://www.voxmlforum.com

[VoxML] Motorola. VoxML 1.2 Language Reference Version 1.0 Rev. 1.5. Http://www.motorola.com, January 2000. See also VoxML Developer Site http://www.voxml.com/voxml.html.

[XML] World Wide Web Consortium. Extensible Markup Language (XML). Http://www.w3.org/XML/