SIEMENS

Information and Communication Mobile

SRIT

Position Statement for Multi-Modal Access

26.11.2001

Authors:

Nathalie Amann, SRIT
(E-Mail: Nathalie.Amann@SRIT.siemens.fr)

Laurent Hue, SRIT
(E-Mail: Laurent.Hue@SRIT.siemens.fr)

Klaus Lukas, Siemens AG, ICM MP TI 4
(E-Mail: Klaus.Lukas@mch.siemens.de)

Content:

1 Introduction and Scope of the Document

The broad application of voice enabled services arises a urgent demand for multi-modal input capabilities for a convenient user interaction. In this document a proposal for such a multi-modal concept is given as a discussion basis for the emerging W3C Multi-Modality Group. This position statement is a consolidated approach of the Siemens Mobile Device and Voice Portal product side, represented by SRIT (Siemens Réseaux Informatique et Télécommunications). Therefore it reflects and covers both views, the device side as well as the server side.

2 Scenarios and Requirements for Multi-Modal Access to Internet Services

A generic model for multi-modal interaction has to cover the major scenarios that can occur for voice-enabled access to Internet services. It is assumed for the following explanations, that the voice interaction is performed using a VUI (Voice User Interface) browser, e.g. a VoiceXML interpreter, whereas the visual interaction is performed on basis of a GUI (Graphical User Interface) browser. The expression " GUI browser" comprises the existing HTML/WML browsers as well as future browsers with advanced markup languages like XHTML.

The main differentiation criteria of the scenarios is the location of the VUI browser, the GUI browser and the speech recognition engine (parts) itself. Figure 1 gives an overview about the identified scenarios to be covered.

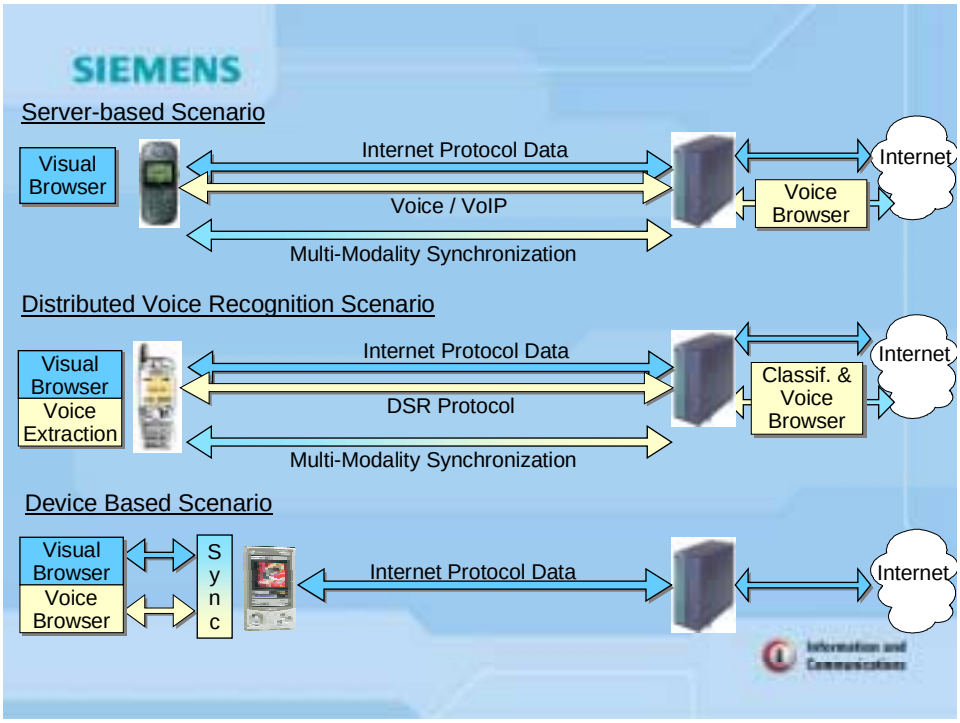

Figure 1: Scenarios for Multi-Modal Access to Internet Services

2.1 Server-Based Scenario

The server-based scenario represents the classical scenario for network-based voice services, where the VUI browser is located on a server. The visual interactions are performed by a GUI browser on the device itself. In this scenario, a multi-modal access requires a network-based synchronization between the visual client operations and the audio server actions.

2.2 Distributed Speech Recognition Scenario

There are two distributed speech recognition scenarios to be considered:

- The feature extraction on device side and the further processing for the recognition itself on server side. This is basically the scenario as discussed in ETSI Aurora. In this scenario the GUI browser is on the client side and the VUI browser on server side. Here, also a network-based synchronization between the device and the server side is required.
- Even if currently less important, the following scenario should be taken into account due to the increasing calculation power of future mobile devices. The VUI browser is in this scenario located on a mobile device itself and forwards speech recognition requests to a server implementation. The speech recognition engine can either be distributed similar to the Aurora approach or can be located on the server e.g. with a VoIP connection. Here, a synchronization of the local GUI browser with the VUI browser is a local matter of control and can be handled on the device without network based synchronization.

2.3 Device-Based Scenario

Finally the device-based scenario covers future trends for more powerful devices, where the browsing functionality for both, visual and voice input/output will be on the device. This scenario fits best to the "Browser paradigm", where a browsing or display facility is located at the end device and the content is located somewhere in the Internet, both linked via Internet communication protocols. Both, the visual and the voice browser will be directly on the mobile device. A synchronization can be handled on the mobile device itself. As this requires a lot of resources and calculation power on the mobile device, this scenario is aimed more at PDAs or 3G-devices rather than on conventional mobile phones.

2.4 Requirements

From our point of view a favorably multi-modal approach has to fulfil the following requirements for a broad application and acceptance:

- The architecture for the multi-modal approach has to be applicable to all scenarios described above
- It has to cover a broad variety of mobile devices (mobile phones, smart phones, PDAs)
- The concept has to be applicable also for 3G devices and networks, but not solely for those. An application for current devices shall also be possible
- A smooth migration of existing systems (e.g. VoiceXML-based Voice Portal Products) and standards (HTML, VoiceXML) shall be possible

3 Concept for Multi-Modal Access

3.1 Architectural Model

The multi-modality approach proposed makes an architectural separation of the visual browsing system and the voice browsing system by defining a synchronization and control model between both systems (see figure 2).

The visual browsing system comprises the GUI browser and a control instance called GUI browser agent. The agent concept guarantees independence of particular browser implementations. The integration into visual browsers can be done by specific PlugIns, by COM oriented approaches or by a direct integration of the agent into the browser.

The voice browsing system consists of the VUI browser (the subsequent speech recognition system and the Text to Speech system) and its application called the VUI browser agent which is in charge of controlling connected GUI browser agents. The VUI browser (e.g. VoiceXML Browser) controls all events and actions regarding speech input and output. Depending on the scenario, the voice browsing system is located on the server side or on the device side. The usage of capability lists allows the devices to decide, whether they can handle voice interactions itself or whether they have to forward it to more powerful or dedicated server engines.

Synchronization will be performed between the GUI browser agent and the VUI browser agent. For providing interoperability between multi-vendor GUI browser agents and VUI browser agents, a synchronization protocol has to be standardized (e.g. based on SIP). Events arising from voice interaction will be sent from the VUI browser agent to the GUI browser agent as Voice Event Notifications via the synchronization protocol. Visual interaction events will be handled vice versa.

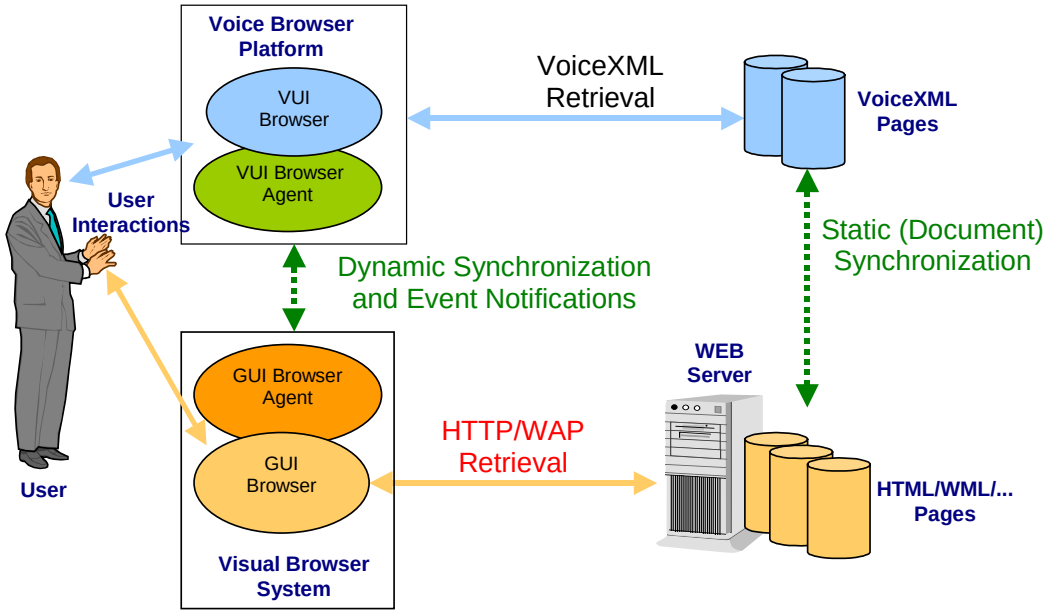

Figure 2: Multi-Modal Access Architecture

3.2 Synchronization and Event Notifications

3.2.1 GUI/VUI Browser Synchronization

The synchronization between the visual part and voice part of the interaction can be split up into a “static” and a “dynamic” synchronization (see figure 2).

- Static synchronization: HTML document link extension referencing to the belonging VoiceXML document.
- Dynamic synchronization: browsing events and user events (manual action with mouse or keyboard, voice action).

3.2.2 Static Synchronization

3.2.2.1 "META tag" Solution

The simple solution used in the SRIT Voice Enable Web Browsing prototype was to used the HTML tag <META> (http://www.w3.org/TR/html4/struct/global.html#adef-scheme) to provide all the required information to link the HTML document with its associated VoiceXML document (see also sample in chapter 7). This anchors the voice browsing information into the HTML page.

Example 1:

```
<HTML><HEAD>
<META name="VoiceXML-URI" content="www.my_site/voice_xml/index.vxml">
</HEAD></HTML>
```

In the example below, the META tag indicates not only the URI of the associated VoiceXML document, but also the ASR capabilities (tier, language, etc.) and TTS capabilities (TTS engine languages, etc.) that are required for being able to browse the VoiceXML page. The list of META information characterizing the complexity of the VoiceXML page is for further study and should be standardized.

Example 2:

```
<HTML>
<HEAD>
        <name="VoiceXML-URI" content="www.my_site/voice_xml/index.vxml">
        <META  name="ASR_tier content="tier3">
        <META  name="ASR_language" content="en-UK,pt-BR">
        <META  name="TTS_language" content="en-UK,pt-BR">
</HEAD>
</HTML>
```

3.2.2.2 LINK Tag Solution

A choice of the LINK tag to anchor the voice browsing information into the HTML page is a standardized approach. The tag LINK in the HTML specification is clearly dedicated to handle such issue. Nevertheless also in this solution, the LINK tag has to consolidated to introduce new media type (voice browser) and a way to provide voice platform capabilities requirement information. The attribute style to provide these capabilities could be a solution.

Example 3:

```
<LINK media="voice" type="text/vxml" href="www.my_site/voice_xml/index.vxml"
style="ASR_tier:3;ASR_language:en-UK;TTS_language:en-UK">
```

The creation of a new tag in current markup languages in order to link visual action to voice action can also be considered.

To link a HTML document to a VoiceXML script it is proposed to push an URL in the HTML browser with “voice_url” action in the VoiceXML script.

Example:

```
<?xml version="1.0"?>
<vxml version="2.0">

<var name="nav_action" expr=""""/>
<var name="nav_value" expr="""" />
<var name="nav_htmlId" expr="""" />

<form>
   <block>
        <audio src="../Audio_files/intro.wav" />
        <goto next="#info"/>
   </block>
 </form>

<form id="info">
  <field name="navigation">
        <grammar src="../Grammars/mobile-world.grammar#VoiceNav" />
        <filled>
        <if cond="navigation=='a35'">
             <assign name="nav_action" expr="'voice_url'" />
             <assign name="nav_value" expr="'http://webserver/MobilePhone/a35.htm'" />
         <elseif cond="navigation=='c30'"/>
             <assign name="nav_action" expr="'voice_url'" />
             <assign name="nav_value" expr="'http://webserver/MobilePhone/c30.htm'" />
         </if>
        <goto next="#push_action" />
        </filled>
   </field>
 </form>

<form id="push_action">
   <object name="action" classid="speechobject://test_speechObject.SOVoiceAction">
     <param name="action" type="java.lang.String" expr="nav_action" />
     <param name="htmlId" type="java.lang.String" expr="nav_htmlId" />
     <param name="value" type="java.lang.String" expr="nav_value" />
   </object>
 </form>

</vxml>
```

3.2.3 Dynamic Synchronization

To be able to modify the state of a HTML document, the linked VoiceXML scenario must be able to identify HTML objects (A, MAP, INPUT, SELECT, TEXTAREA, BUTTON, and FORM). The solution is to use the HTML identification attribute “id”, where the name value must be unique in the HTML document (http://www.w3.org/TR/html4/struct/global.html#h-7.5.2).

The “id” attribute assigns a unique identifier to an element (which may be verified by an SGML parser).

Example:
The following paragraphs are distinguished by their “id” values:

<P id="myparagraph"> This is a uniquely named paragraph.</P>
<P id="yourparagraph"> This is also a uniquely named paragraph.</P>

The “id” attribut has several roles in HTML (See HTML RFC documentation). The interesting role is here, that it can be used for general purpose processing by user agents (e.g. for identifying fields when extracting data from HTML pages into a database, translating HTML documents into other formats, etc). With the help of this identification, it is easy to link voice events to HTML objects.

Example:
Some exemplary VoiceXML events are listed below:

- voice_click id=link1 (Voice click event on the HTML object, whose id is “link1”)
- voice_input id=input_departure_form value=”Paris” (Voice input event on the HTML object, whose id is “input_departure_form” and whose field value is “Paris”)
- voice_focus id=input_departure_form (Voice focus on the HTML object, whose id is “input_departure_form”)

3.2.4 Synchronization issues

The synchronization protocol must be able to handle multi HTML documents in the same session and be able to know the active HTML document and its linked VoiceXML session.

3.2.5 Voice Event Notification

The issue here is to find solutions based on VoiceXML markup language standard (http://www.w3.org/TR/voicexml20/) to integrate in the VoiceXML scenario the voice navigation events .

Voice Event Notifications are shown below:

- Activate HTML links or button form on speech recognition events (“voice_click” action)
- Fill HTML form with specific values from speech recognition events (“voice_input” action)
- Set focus on HTML elements from speech recognition events (“voice_focus” action)
- Select an item in a HTML combo box (“voice_select” action)
- Select a HTML check box or a HTML radio button (“voice_check” action)
- Push a URL in a HTML browser (“voice_url” action)

- Control classical navigation actions of HTML browser from speech recognition events (“voice_navigation” action with the following value: Back, Forward, Stop, Refresh, Home, Print, Close)

A VoiceXML implementation platform may have platform-specific functionality that an application wants to use, such as speaker verification, native components, additional telephony functionality, and so on. Such platform-specific objects are accessed using the OBJECT element, which is analogous to the HTML <OBJECT> element

The OBJECT element will be used to specify "builtin" voice events invoking synchronization activities with the browser agent.

Example:
<object name=”voice_click” classid=”builtin://voice_click”>
<param name=”id” expr=”country”>
</object>

3.2.6 Visual Event Notification

These Event Notifications might be for synchronizing HTML and VoiceXML scenario using HTML events:

- Activate VoiceXML Dialog on HTML events (“mouse focus” action)

Example:
A user loads an URL, the VoiceXML scenario associated to this HTML page is played. The user clicks on a link in the HTML page. This HTML event stops the current VoiceXML scenario and launches the new VoiceXML scenario corresponding to the new HTML page.

Remark:
The voice flow delay can influence the synchronization between the display of the HTML page and the beginning of the voice flow.

3.3 Walk Through

3.3.1 Server Based Scenario

In the following a walk-through is described for a server-based scenario (see figure 3).

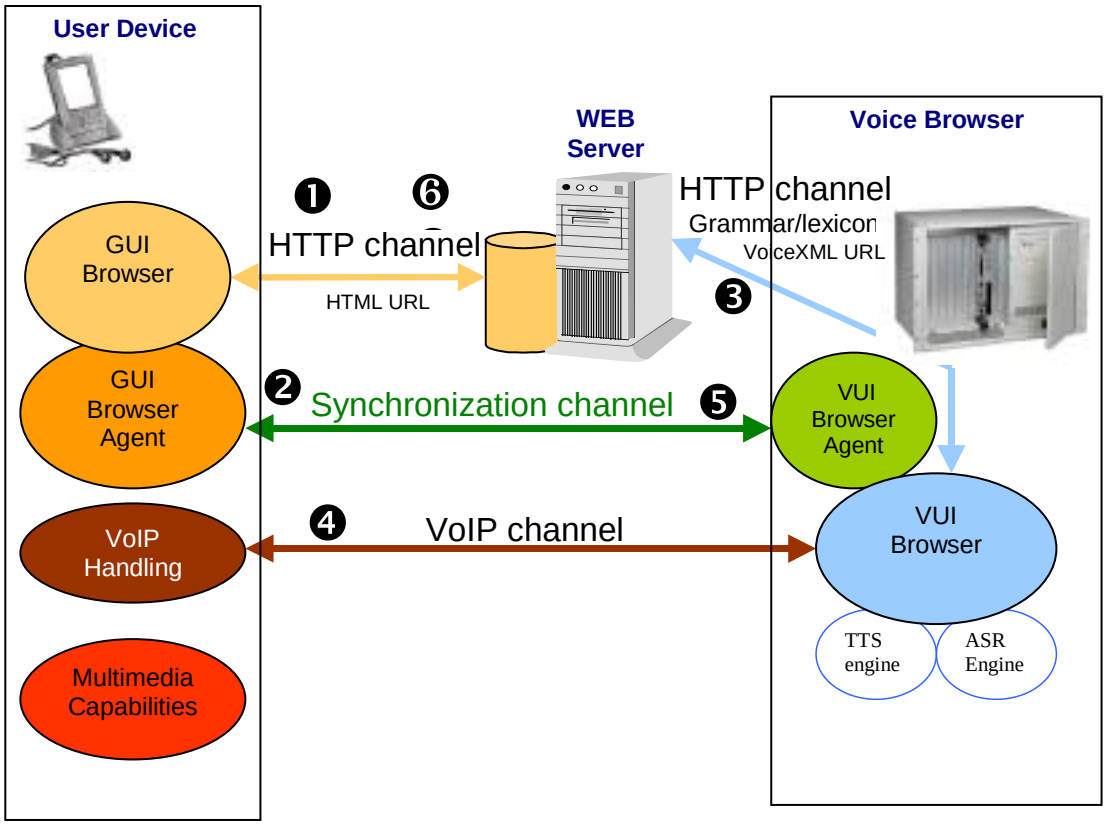

Figure 3: Walk Through for Server Based Scenario

❶ The GUI browser loads an URL from a WEB server
❷ The GUI browser agent checks the capability list (“VoiceXML-URI” value of the content parameter in the LINK tag) and forwards the VoiceXML URI to the VUI browser agent
❸ The VUI browser loads and interprets the VoiceXML script of the URI and the associated grammars
❹ The user can speak to the VUI browser and listen to it
❺ The voice browser triggers voice recognition events to the GUI browser via the GUI browser agent
❻ The GUI browser retrieved the URL linked to voice events (voice click, voice navigate, etc.)

3.3.2 Distributed Speech Recognition Scenario

This scenario is the same as the Server Based Scenario, with the exception that the VoIP connection is replaced by a DSR based protocol, DSR client and server component.

3.3.3 Device Based Scenario

In this scenario, both the visual browsing (HTML/WML) and the voice browsing (VoiceXML) run on the device (see figure 4).

It is up to the terminal to decide whether it has the capabilities to browse the VoiceXML scenario. The device can determine the complexity of the Voice script (VoiceXML) to be downloaded with the multi-valued "capability" attributes passed in the META (or LINK) tag of the visual markup language page (HTML/WML). The values for this "capability" parameter should be standardized (e.g. within W3C CC/PP) and reflect the complexity of the voice page from an ASR and TTS point of view:

- ASR complexity (Yes/No, Digit, word spotting, multiword spotting, …).
- TTS complexity (presence of TTS, languages to be supported, …).

Depending on its capabilities:

- If the device is not able to browse the scenario, then the work flow of the Server Based Scenario can be used.
- If the device is able to browse the scenario, then the device downloads the voice script (VoiceXML) and associated grammars and starts the voice browsing (see Device based scenario).

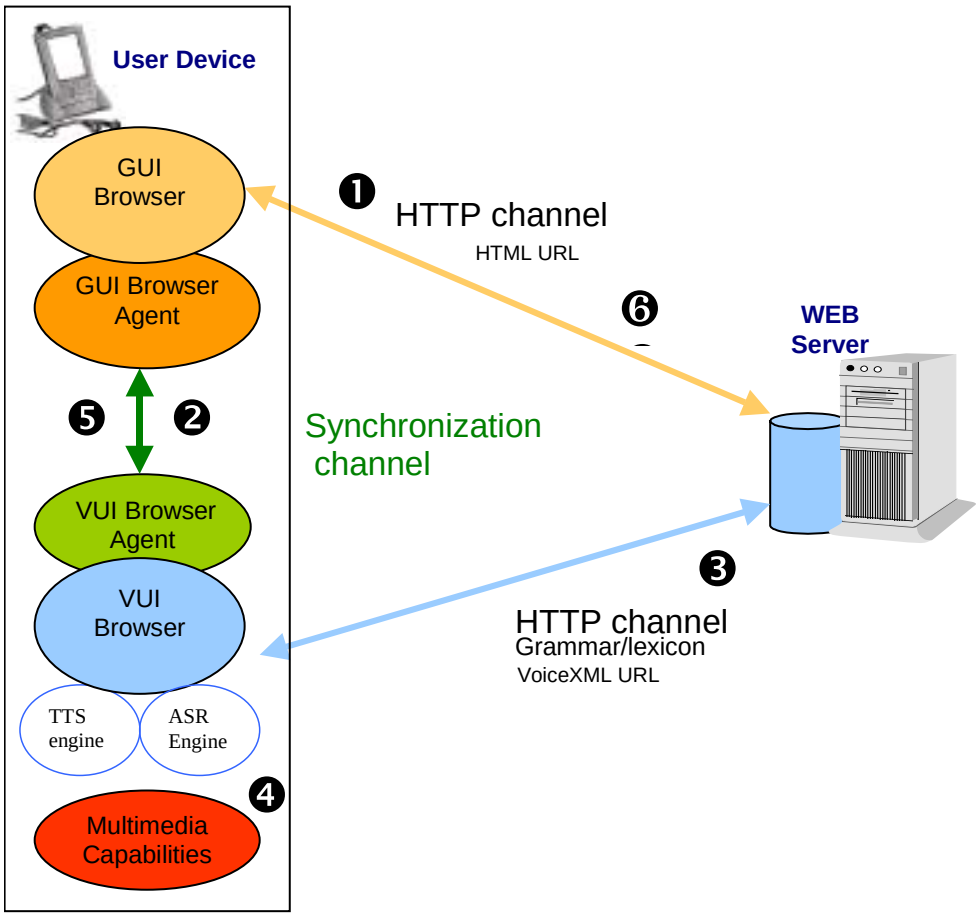

Figure 4: Walk through for Device Based Scenario

❶ The GUI browser loads an URL from a WEB server
❷ The GUI browser agent checks the capabilities list (“capability” attributes in the META (or LINK tag) and forwards the VoiceXML URI to the VUI browser agent on the device
❸ The VUI browser loads and interprets the VoiceXML script of the URI and the associated grammars
❹ The user can speak to the VUI browser and listen to it
❺ The VUI browser triggers voice recognition events to the GUI browser via the VUI and GUI browser agent
❻ The GUI browser retrieves the URL linked to voice events (voice click, voice navigate, etc.)

4 Evaluation of Concept

4.1 Concept Benefits

The proposed concept allows the integration of multi-modal functionality to existing systems and standards with low impact. It requires only small extensions to current markup languages for tying the voice actions to the visual content. Therefore the changes to the standards and the product based on the standards will be rather low. This allows a smooth migration starting from existing systems and products.

The requirement for covering the different scenarios is fulfilled. In case of the server-based scenario and the first DSR scenario a network-based synchronization will be applied. For the device-based scenario and the second DSR scenario even a local synchronization may take place. In best case, both components, the visual browser and the voice browser will be merged to one unit with an internal synchronization. Applying the multi-modal approach described allows a huge flexibility and platform independence. As the proposed approach covers all three scenarios, a high scalability is given for deploying multi-modality approaches to a broad variety of systems even covering future device oriented speech recognition systems and distributed architectures.

Finally the "Dialog Requirements for Voice Markup Languages" [1] have been taken into account for establishing the proposed architecture.

4.2 Impacts on existing Standards

The impacts on existing standards are low. The visual markup languages (e.g. HTML, WML, XHTML) have to be extended by an URI Extension to link to the respective VoiceXML Page.

The VoiceXML specification can be used as is and requires only an extended profiling for the OBJECT functionality [2].

4.3 Requirement for new Standards

For the synchronization between the GUI browser agent and VUI browser agent a protocol has to be defined. The protocol has to comprise the dynamic synchronization elements as well as the visual and voice notification events.

5 Feasibility

The feasibility of the proposed architectural model has been verified by a demonstrator available at SRIT. The demonstrator is based on the Siemens VoxPortal© product and shows the multi-modal approach in the context of a server-based scenario.

6 Proposed Issues to be Covered within the Multi-Modality Working Group

A main issue of the multi-modality working group should be the definition of a common architectural model for multi-modal systems. This position statement is intended to serve as a discussion basis for the emerging W3C Multi-Modality Group. The extension of existing standards, the consideration of existing systems and the flexibility for covering a broad range of possible scenarios are further important aspects to be discussed within the group.

Furthermore the provision of consistent visual and voice scripts will be a crucial task for authoring tools. Hereby, proposals are required taking single authoring approaches into account for a migration path from the current multiple authoring schemes to a future unique source of content.

7 Examples

- HTML-Page example


```
<HTML>
<HEAD>
<TITLE>my-siemens</TITLE>

<META name="VoiceXML-URI" content="http://WebServer/MobilePhone/voicexml/SL45.vxm">

<META content="text/html; charset=windows-1252" http-equiv=Content-Type>
<META content="MSHTML 5.00.2314.1000" name=GENERATOR>
</HEAD>

<FRAMESET border=0 frameBorder=0 frameSpacing=0 rows=*,450,*>
        etc.
</FRAMESET>
</HTML>
```

- VoiceXML-Page example

```
<?xml version="1.0"?>
<vxml version="2.0">

<var name="nav_action" expr=""""/>
<var name="nav_value" expr="""" />
<var name="nav_htmlId" expr="""" />

<form>
<block>
<audio src="../Audio_files/chimes.wav" />
<goto next="#info"/>
</block>
</form>

<form id="info">
  <field name="navigation">
    <audio src="../Audio_files/2-3-2-01.wav" />
    <grammar src="../Grammars/PhoneNav.grammar#PhoneNav" />

    <nomatch><audio src="../Audio_files/2-3-2-12.wav" /><reprompt /></nomatch>
    <noinput><audio src="../Audio_files/2-3-2-13.wav" /><reprompt /></noinput>
    <help><audio src="../Audio_files/2-3-2-02.wav" /></help>
    <catch event="cancel"><audio src="../Audio_files/2-3-2-12.wav" /><reprompt /></catch>

    <filled>
    <if cond="navigation=='details'">
       <assign name="nav_action" expr="'voice_url'" />
       <assign name="nav_value" expr="'http://webserver/MobilePhone/SL45_details.htm'" />
   <elseif cond="navigation=='home'"/>
       <assign name="nav_action" expr="'voice_url'" />
       <assign name="nav_value" expr="'http://webserver/MobilePhone/mobile-world.htm'" />
   <elseif cond="navigation=='survey'"/>
      <assign name="nav_action" expr="'voice_url'" />
      <assign name="nav_value" expr="'http://webserver/MobilePhone/survey.htm'" />
    </if>
   etc…
    <goto next="#push_action" />
    </filled>
  </field>
</form>

<form id="push_action">
   <object name="action" classid="speechobject://test_speechObject.SOVoiceAction">
    <param name="action" type="java.lang.String" expr="nav_action" />
    <param name="htmlId" type="java.lang.String" expr="nav_htmlId" />
    <param name="value" type="java.lang.String" expr="nav_value" />
   </object>
</form>
</vxml>
```

- Grammar example

```
PhoneNav[
 (?technical data)~1.0 { <navigation "details"> }
 home~0.5 { <navigation "home"> }
 (?phone survey)~1.0 { <navigation "survey"> }
 ]
```

8 References

[1] "Multimodal Requirements for Voice Markup Languages", W3C Working Draft, 10th July 2000, http://www.w3.org/TR/multimodal-reqs

[2] "Voice Extensible Markup Language (VoiceXML) Version 2.0", W3C Working Draft, 23th October 2001, http://www.w3.org/TR/voicexml20