A Multimodal Interface for the Discovery and Invocation of Web Services

CÁSSIO V. S. PRAZERES, MAYCON L. M. PEIXOTO

e-mail: prazeres@dcc.ufba.br, mayconleo@gmail.com

Universidade Federal da Bahia (UFBA)
Departamento de Ciência da Computação
Avenida Adhemar de Barros, Campus de Ondina
Salvador - BA

Universidade de São Paulo (USP)
Instituto de Ciências Matemáticas e de Computação
Avenida Trabalhador São-carlense, 400 - Centro
São Carlos - SP

Abstract. The World Wide Web has been enhanced with many types of online services such as banking access, e-commerce, e-health, e-learning, etc. This variety of services, distributed around the world, can be accessed from any place at anytime. However, in some situations a user may face difficulties in the use of the traditional mouse and keyboard input devices to search and access services through the Web. In this paper we propose the use of multimodal interfaces for the discovery and invocation of Web Services. Our approach, called Multimodal Interface for Discovery and Invocation of Web Services (MIDIWS), transforms users requests made in XHTML + VoiceXML into requests specified as SOAP messages. These requests are sent to a UDDI repository which returns a service or a set of services also by means of SOAP messages. These messages are transformed back into XHTML + VoiceXML and, then, presented to the user.

Keywords: Multimodal interface, VoiceXML, Web Services.

(Received March 6th, 2013 / Accepted September 14th, 2013)

1 Introduction

The grow in the number of users in the World Wide Web has been accompanied by a growth in the diversity of applications in areas such as e-commerce, e-health, e-learning, to name a few. Many of these applications are made available by means of Web Services.

The W3C (World Wide Web Consortium) defines Web Services[1] as "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards".

In many situations, the aim is to make services available to end-users by means of web applications. When this is the case, the discovery and the invocation of available services can be made by means of search engines interfaces in which the user provides keywords related to the desired service.

In order to spread the access to your Web Services,

[1] http://www.w3.org/TR/ws-gloss/

some companies may want to make available your services to be accessed by means of other modalities as not only mouse and keyboard (as voice modality for example). Users can provide the keywords via voice input, for instance, supported by speech recognition technology, as envisioned by Dettmer [2].

Two special initiatives from the W3C point to many challenges that some users face in accessing the Web: the *Web Accessibility Initiative* [17] and the *Device Independence Working Group* [18]. The frist initiative points to "challenges such as impaired vision, limited motor capabilities or difficulties with perception that diminish their ability to use the Web" [21]. The second one point that "there are Web users who face challenges because of device or network limitations, including small screens, restricted keyboards and lower bandwidth" [21].

The approach proposed in this paper can address these two types of challenges pointed by the W3C [21]. This approach uses multimodal interface for the discovery and invocation of Web Services. Our approach is called MIDIWS (**M**ultimodal **I**nterface for **D**iscovery and **I**nvocation of **W**eb **S**ervices).

In MIDIWS, the user interactions in any modality (as speech or gesture, for example) can be mapped to SOAP [20] (Simple Object Access Protocol) messages that are sended to UDDI [9] (Universal Description, Discovery, and Integration) registries. In order to demonstrate our approach, we implement a case study by using speech modality. This case study focus on the case that the user who access the Web face challenges because of device limitations, as, for example, restricted keyboards.

MIDIWS has a SOAP engine and an agent that interact with multimodal interfaces (customer side) and a UDDI registry (server side). In a first step, the SOAP engine gets requests in XHTML + VoiceXML and transforms to requests in SOAP messages. Second, the agent receive these SOAP messages and send to the UDDI registry. The UDDI sends back to the agent the response (SOAP messages) about the search for a service in a third step. Fourth, the agent sends these response to the SOAP engine that transforms back to XHTML + VoiceXML and presents to the user.

In order to provide multimodalities for discovery and invoke Web service, we present an engine that make transformations in XML documents (XHTML + VoiceXML to SOAP and vice-versa) and an agent that works as a mediator between the engine and the UDDI registry. We implement an infrastructure based on an Web application container (Apache Tomcat), Web services framework (Apache Axis2) and UDDI registry (Apache jUDDI) integrated with our SOAP engine and agent to develop and to test the MIDIWS approach.

In the remainder of this paper, Section 2 reviews concepts of multimodal interfaces; Section 3 reviews current infrastructure and technologies for Web Services; Section 4 introduces MIDIWS; Section 5 details a scenario in which a "rent a car service" is made available to users; Section 6 presents an evaluation, based on two important techniques of usability evaluation, of our case study; Section 7 discusses related work and Section 8 presents our final remarks.

2 Multimodal Web Applications

Multimodal Web applications are those in which it is possible to interact with a traditional Web browser via synthetic speech, keypads, pointing devices, prerecorded audio, plain text and displays.

The technologies SALT (Speech Application Language Tags) and X+V (XHTML + VoiceXML) represent the first steps in building mechanisms of speech-enabled Web [5]. SALT and X+V are both tag-based languages created to enable multimodal applications by adding voice tags to HTML/XHTML.

In the work presented in this paper we use X+V language for a number of reasons: it is a standard proposed by the W3C, there is a community of developers highly trained in VoiceXML, and several supporting tools are available. Moreover, it is supported by some browsers (such as the Opera), thus enabling developers to immediately test the interfaces produced.

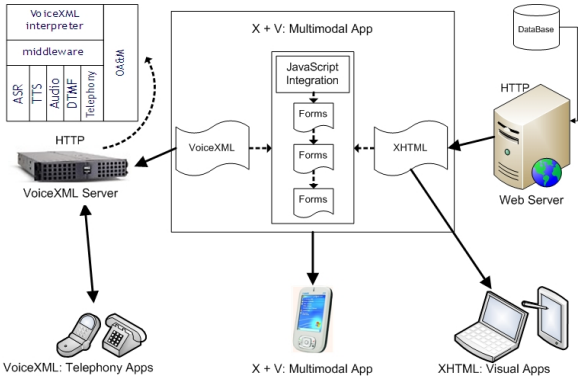

Figure 1: Model Architecture for Voice Browser Systems

As shown in Figure 1, the multimodal applications using standard X+V allows a natural migration of voice applications based on VoiceXML, generally used in telephony applications, to applications that include the voice element to be used with the layout language XHTML by means of Javascript Sync. It becomes an

important feature to enable voice applications to perform discovery of web services, with the intention of reaching external resources from limited devices.

3 Web Services and UDDI

Web Services are software systems that are identified by means of a URI and that are specified by a description based on XML documents. The XML description permits the interaction among several systems. XML documents are also used to communicate with others systems through messages exchange using Internet protocols [19].

Web Services are based on standard technologies incluing XML, SOAP, WSDL [22] (Web Service Description Language) and UDDI [9] (Universal Description, Discovery, and Integration). The interaction among an application based on Web Services and a Web Service uses the SOA (Service Oriented Architecture).

The XML has an important function in the Web Services realization, because the involved technologies (WSDL, SOAP and UDDI) are based on the XML language.

The language for service description (WSDL) is used to describe a Web Service by specifying it localization and by describing the operations provided by the service. WSDL documents are XML documents that provide enough information about how to interact with the Web Service.

SOAP is a communication protocol and a codification format based on XML that has a function of realize communication among applications.

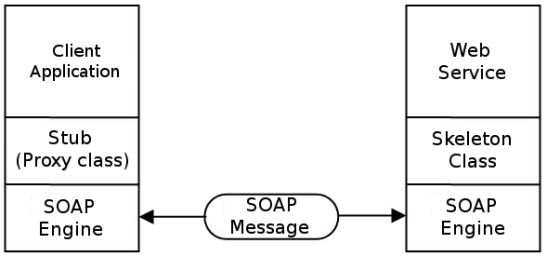

Figure 2: Communication infrastructure for Web Services.

The Figure 2 (adapted from [6]) shows an infrastructure of basic communication that permits the use of Web Services by using SOAP messages exchange. The objects from the customer side (service requester) uses the services by means of a proxy class that ***imitate*** the method calls of the Web Service. Thus, the application developers use this proxy class instead of directly write SOAP messages. The proxy class manages the construction, sending and receiving of SOAP messages.

The UDDI is a registry of domain public – service providers can register their services and service requesters can search for a desired service. The information about registered services (UDDI) work as an information set about all registered Web Services, making possible the discovery of services providers and services.

4 MIDIWS

MIDIWS, **M**ultimodal **I**nterface for **D**iscovery and **I**nvocation of **W**eb **S**ervices, combines the use of multimodalities for interact with Web applications and the Web services infrastructure based on SOA architecture.

4.1 MIDIWS Architecture

The MIDIWS architecture is shown in Figure 3. The overall implementation is deployed in a Web application container (Figure 3: Apache Tomcat).

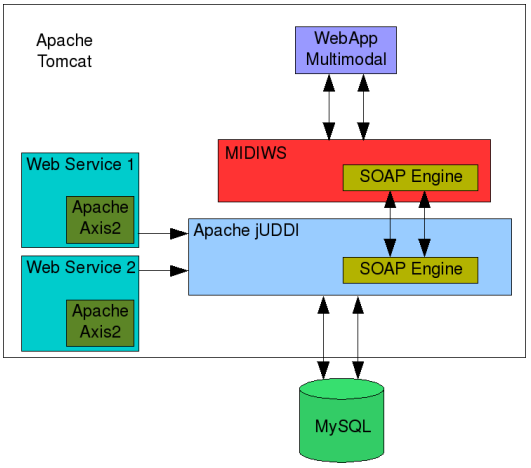

Figure 3: MIDIWS Architecture

The UDDI implementation used is the jUDDI registry (Figure 3: Apache jUDDI), which is deployed on the Web application container (Figure 3: Apache Tomcat).

In the MIDIWS architecture, we use a relational database (Figure 3: MySQL) integrated to the jUDDI registry to store all UDDI information relative to Web Services.

As far as the MIDIWS architecture is concerned, the important information regarding the Web services is that stored in the UDDI repository. For the sake of illustrating our proposal, two Web services are also deployed in the Web application container using the Apache Axis2, which is a framework to develop Web services (Figure 3: Web Services 1 and 2).

Naturally, the Web services do not need to be implemented in the same machine that was implemented the MIDIWS neither in the same machine that was implemented the jUDDI. Meanwhile, as can be viewed in Figure 3 (Web Service 1 and 2), we implement some Web services to make possible that some real Web services can be stored in our jUDDI implementation.

MIDIWS makes use of two SOAP engines: (i) one to send the requests from users to be processed by the jUDDI registry (see Figure 3: MIDIWS), and (ii) the other to send the responses from the jUDDI regitry back to the users (see Figure 3: Apache jUDDI). These communications are made via SOAP messages and work as was presented in Section 3 (see Figure 2).

To experiment with our approach, we have also deployed in the Web application container (Figure 3: Apache Tomcat) a Web application that is integrated with the MIDIWS (Figure 3: WebApp Multimodal). This Web application has a multimodal interface based on visual (mouse and keyboard) and voice modalities and works as explained below.

4.2 MIDIWS Workflow

The MIDIWS architecture follows the 6-step workflow shown in Figure 4. In step (1), the MIDIWS SOAP Engine sends a SOAP message to the UDDI registry. In this step, this SOAP Engine builds a SOAP message with a request to find the service requested. *Document 1* below represents a SOAP message used to inquiry about a *Rent a Car Company* service: lines 5 to 9 in *Document 1* contain the information about the requested service (in this case, the service name).

```
0  <!--        Document 1        -->
1  <!-- SOAP message for Find a Service -->
2
3  <?xml version="1.0" encoding="utf-8"?>
4  <soapenv:Envelope xmlns:soapenv=
     "http://schemas.xmlsoap.org/soap/envelope/">
5             <soapenv:Body>
6    <find_service generic="2.0" xmlns="urn:uddi-
              org:api_v2">
7     <name>Rent a Car Company</name>
8    </find_service>
9   </soapenv:Body>
10 </soapenv:Envelope>
```

In step (2), the UDDI registry responds to the SOAP-Engine, mediated by the Agent shown in Figure 4, with a list of discovered services. Lines 6 to 18 in the *Document 2* below present a list of all discoved services that match the parameters requested.

```
0  <!--        Document 2        -->
1  <!-- List of discovered services -->
2
3  <?xml version="1.0" encoding="UTFlist-8"?>
4  <soapenv:Envelope xmlns:
       soapenv="http://schemas.xmlsoap.org/soap/
envelope/"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema
-instance">
5   <soapenv:Body>
6    <serviceList generic="2.0" operator="jUDDI.org"
                xmlns="urn:uddi-org:api_v2">
7     <serviceInfos>
8      <serviceInfo
          businessKey="13B7B200-41FD-11DD-BB88-
3C09B365B"
          serviceKey="4B874330-41FD-11DD-BB88-
E7D2B7EA6">
9       <name>Rent a Car Company 1</name>
10     </serviceInfo>
11     <serviceInfo
          businessKey="4064C064-6D14-4F35-8953-
2106476A9"
          serviceKey="1A85B7E0-3C89-11DD-B39D-
43AC41B80">
12      <name>Rent a Car Company 2</name>
13     </serviceInfo>
14     <serviceInfo
          businessKey="C0B9FE13-179F-413D-8A5B-
4DB8E5BB2"
          serviceKey="B1B1BAF5-2329-43E6-AE13-
E97195039">
15      <name>Rent a Car Company 3</name>
16     </serviceInfo>
17    </serviceInfos>
18   </serviceList>
19  </soapenv:Body>
20 </soapenv:Envelope>
```

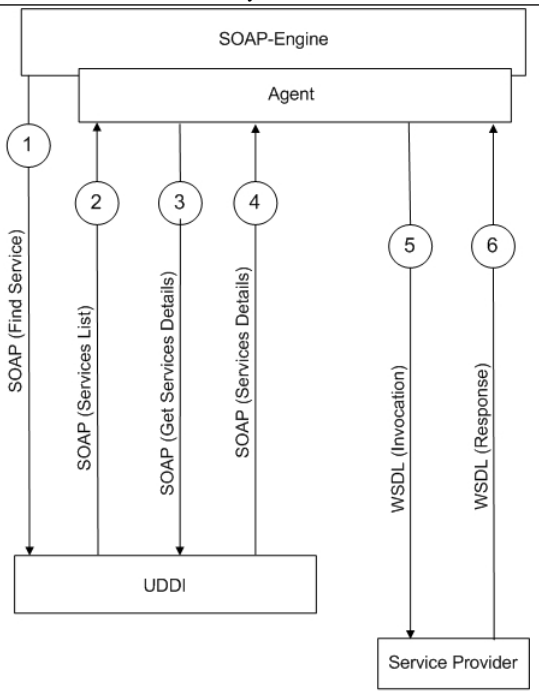

Figure 4: MIDIWS Workflow

In step (3) shown in the Figure 4, a SOAP message is sent from the Agent to the UDDI registry to request all information about the services discovered, as indicated in the lines 6 to 10 of *Document 3*.

```
0  <!--        Document 3        -->
1  <!-- Get details of a discovered service -->
2
3  <?xml version="1.0" encoding="utf-8"?>
4  <soapenv:Envelope xmlns:soapenv=
           "http://schemas.xmlsoap.org/soap
/envelope/">
5   <soapenv:Body>
6    <get_serviceDetail generic="2.0"
                        xmlns="urn:uddi-org:api_v2">
7     <serviceKey>
8      4B874330-41FD-11DD-BB88-FEFE7D2B7EA6
9     </serviceKey>
10   </get_serviceDetail>
11  </soapenv:Body>
12 </soapenv:Envelope>
```

In the step (4) of the workflow shown in Figure 4, the UDDI repository sends a list to the Agent with details about discovered services as shown in *Document 4* below. The resulting information is used by the Agent to, by means of WSDL invocations to the service providers in step (5), obtain further information of their services. The responses, indicated in the step (6) of the workflow, are used by the Agent to choose a service and to preform further transactions.

```
0  <!--        Document 4        -->
1  <!-- Services Details -->
2
3  <?xml version="1.0" encoding="utf-8"?>
4  <soapenv:Envelope xmlns:soapenv=
           "http://schemas.xmlsoap.org/soap/
envelope/">
5   <soapenv:Body>
6    <name> $Name Service$ </name>
7    <description> $Description Service$
</description>
8    <accessPoint> $URLType$ </accessPoint>
9   </soapenv:Body>
10 </soapenv:Envelope>
```

5 A Case Study

We performed a case study which we experimented using the Opera browser. When using the Opera browser, users can enable XHTML + VoiceXML support through steps described at the Opera Voice site[2].

5.1 "Rent a Car" using MIDIWS

We implemented a service allowing a hypothetical company to offer the possibility of a user to rent a car by means of a multimodal interface using XHTML and VoiceXML. A front-end application (see Figure 5) to capture the information is provided to the user. Naturally, this is only an example of a service: it could be any service that a user would like to access via multimodal interfaces.

Customers should fill in a Web form with their interests about the wanted service. As an example, one component of this interface is the step where the city is chosen, which could the one presented in Figure 5. Other typical components of the interface associated with the Rent a Car Service we implemented are: Place to pick up the consumer; Vehicle type: car ou van; Vehicle model: models are linked to the car type; Vehicle class: full or standard; Distance plan; Car insurance; Pick up information: pick up date and pick up time; Return information: return date and return time.

In the case of the interface "City:" (see Figure 5), our implementation provides a component radio button for visual interaction (XHTML) or through voice (VoiceXML). The first case (visual) is shown in *Document 5*.

```
0  <!--        Document 5        -->
1  <!-- XHTML source for "choose a city" -->
2
3  <!-- City -->
4  <div id="divCityId">
5   <b>City:</b>
6   <br/>
7   <input type="radio" name="city"
           id="cityNewYorkId"
           value="New York"/> New York;
8   <input type="radio" name="city"
           id="cityLosAngelesId"
           value="Los Angeles"/> Los Angeles;
9   <input type="radio" name="city"
           id="cityChicagoId"
           value="Chicago"/>Chicago;in the
10  <input type="radio" name="city"
           id="cityHoustonId"
           value="Houston"/>Houston;
11 </div>
```

To provide the voice interface, it is necessary to define a grammar that will recognize the user and ways to help him to fill in the correct information (see *Document 6*).

```
0  <!--        Document 6        -->
1  <!-- VoiceXML source for Recognition
Grammars -->
2
3  <!-- City -->
4  <vxml:field name="voice_field_city"
              xv:id="voice_city" modal=
"true">
5   <vxml:grammar>
6    <![CDATA[
       #JSGF V1.0;
       grammar city;
       public <city> = New York |  Los Angeles |
                       Chicago | Houston;
     ]]>
7   </vxml:grammar>
8   <vxml:prompt> Choose the city </vxml:prompt>
9   <vxml:catch event="help nomatch noinput">
10   For example, say New York.
11  </vxml:catch>
12  <vxml:catch event="help nomatch noinput"
count="2">
13   Please, choose New York, Los Angeles,
Chicago or Houston.
14  </vxml:catch>
15 </vxml:field>
```

If the user is filling out the form by voice, is necessary to carry out a synchronization between the modalities, as shown in *Document 7*.

[2] http://www.opera.com/voice/

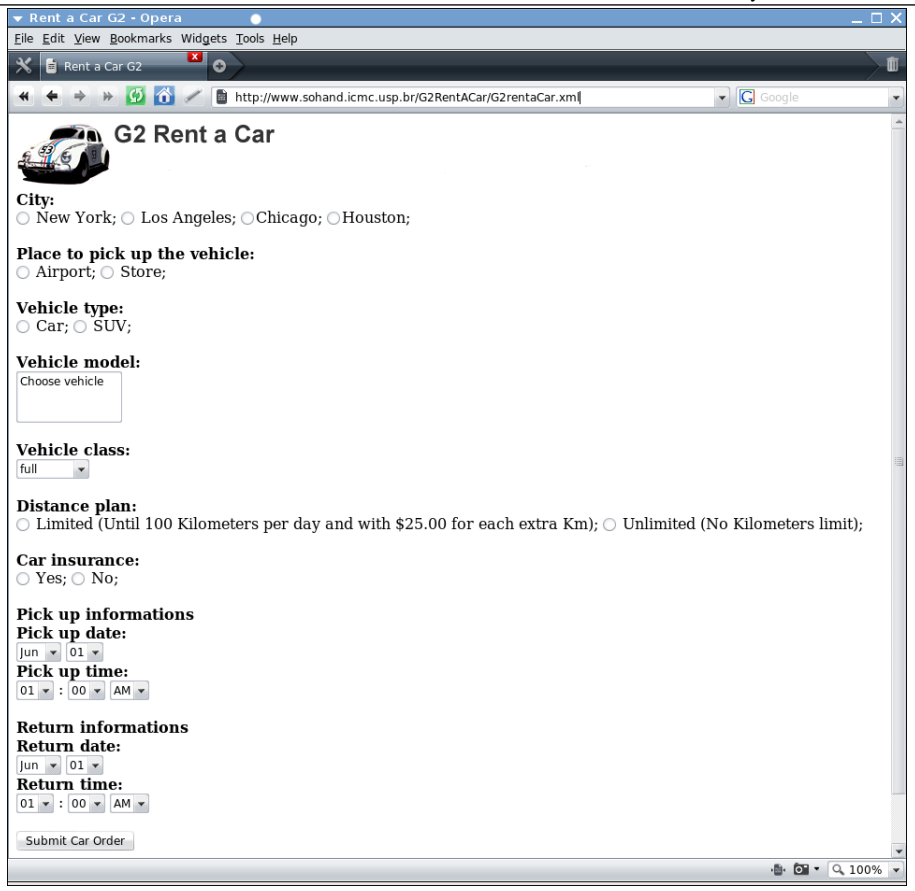

Figure 5: Rent a Car Service

```
0  <!--        Document 7        -->
1  <!-- Sync between XHTML and VoiceXML -->
2
3  <!-- City -->
4  <xv:sync xv:field="#voice_city"
xv:input="city"/>
```

Summarizing our example implementation, when an interface is loaded by the browser, the focus is directed on text field HTML. The aim is to obtain the desired city. The same VoiceXML form represented by "id=voice_city" requires a *prompt* that ask to the user: *Please, Choose the city!*.

5.2 Detailing the case study

We divide a case study execution of the "Rent a Car" service in steps as represented in Figure 6.

First, the user is presented with a form (Figure 6: Web Application Multimodal) to fill in an order to provide the Engine-agent with details about choices his/her rent a car service.

In the second step, the user submits the information (XML: XHTML or VoiceXML) to the SOAP engine (Figure 6: SOAP-Engine) in order to build a list of services corresponding to his criteria.

Third, the Agent (Figure 6: Agent) finds a list of "Rent a Car" services in UDDI. For each service available:

- The Agent requests a description of how to communicate with the service found (WSDL);
- The Agent requests a list of details about services;
- The UDDI registry (Figure 6: UDDI) returns the requested details.

In the fourth step, the Agent joins all descriptions submitted by UDDI and selects the best service. The best selection criteria is not implemented in our current version. We will implement smart agents, based on OWL-S ontology, to perform selection based on the semantics of the Web service.

Finally, the confirmation data of the service done is presented to the user and a confirmation message is presented to the user. When voice is used, the correspond-

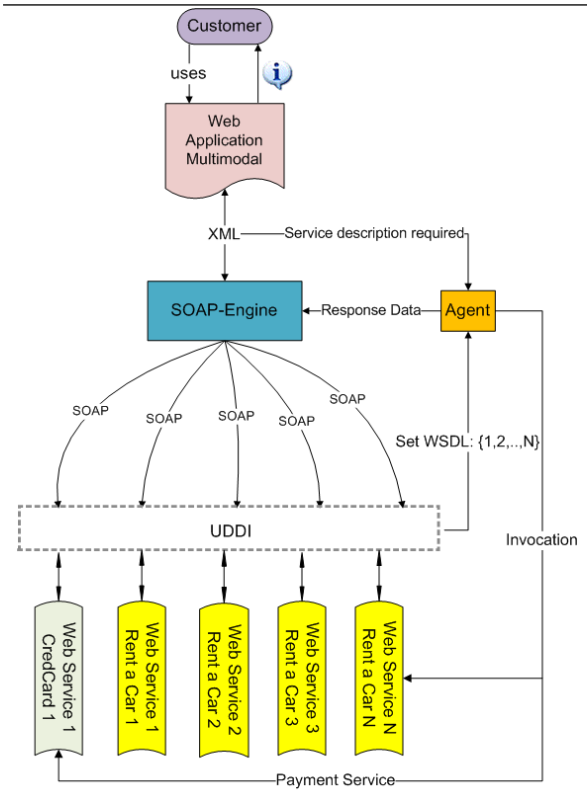

Figure 6: MIDIWS Workflow: a Case Study Execution.

ing message is: $< prompt > Successful\ service! < /prompt >$.

At the end of these steps, if no rent a car company service can be found, the user should be presented with an appropriated message.

6 Evaluating the "Rent a Car" Service

We evaluate our case study of "Rent a Car" Service (see Figure 5) by using two important techniques of usability evaluation: i) the Heuristic Evaluation [8], which encompass several features of the user interaction; and ii) the Think Aloud Evaluation [14], which has focus on evaluating usability of interactive learning systems or even websites.

In the Heuristic Evaluation, we use the ten heuristics of Nielsen [8] and we made general analysis of the interface, identifying gaps and assignment of one of the five levels of severity that guide the development on the priority and urgency of a solution.

In the Think Aloud Evaluation, we bring a real user to try to rent a car by using the MIDIWS and this user reports step by step all that he/she intends to do, why he/she to do, and what he/she expect of system response.

The user interface of the "Rent a Car" case study does not present a high complexity of interaction, as is the completion and submission of a Web form by voice to rent a car. Still, during the evaluation process were identified some deficiencies in the application.

The Heuristic Evaluation has spent two hours and was required to install and configure the Opera browser for the MIDIWS could be used and evaluated with voice interaction.

Aspects of the system interface do not have serious flaws, but there are things that can be improved, such as: the grouping of similar data by using edges and titles for such; and the increasing in vertical spacing between the titles and the boxes, because this area is currently very little and overlap the graphics.

The Think Aloud Evaluation pointed to one problem: the evaluator did not perceives whether the task had been completed, since it is not given the proper feedback when the Web form of rent a car is submitted.

The grouping of similar data in the Web form is a trivial problem when using interaction by mouse and keyboard. However, considering only the interaction by voice, it is important to further evaluation whether the grouping of similar data can help the user interaction.

Concluding, there are some errors found during the evaluation and must be repaired to make the interaction more efficient and increase the usability of the application ("Rent a Car" Service).

7 Related Works

Several researchers investigate approaches to use multimodal interaction in Web applications(e.g. [2],[1],[4],[11],[16],[7]). These authors present and discuss several implementations, tools and technologies for multimodal interactions over the Web and over mobile technologies. In MIDIWS, we present an infrastructure to discover and invoke Web services by using multimodal interactions.

Dettmer [2] remarks that mobile technologies, as mobile phone, provide a universal platform for accessing on-line services. He advocates that VoiceXML can be combined with visual technologies (as XHTML) to access on-line services by both voice and graphics. In MIDIWS, we also combine VoiceXML and XHTML in the user side. We integrate VoiceXML and XHTML with an infrastructure of Web services to provide discovery and invocation of on-line services.

Ramakrishnan et. al. [11] [16] propose a voice browser system that provides access to the World Wide Web to persons with visual disabilities. These systems permit the browsing of hypertext Web documents via

audio by transforming HTML document in audio documents (VoiceXML). MIDIWS assumes that the user interface is on XHTML + VoiceXML and transforms users requests into SOAP messages to make Web service discovery and invocation.

Honkala and Pohja [4] discuss that providing XHTML + VoiceXML in multimodal interfaces is not trivial since each modality has to be authored separately. The authors propose to use XForms as the user interface description language, because it is independent of modality. In MIDIWS, we use XHTML + VoiceXML only to test our approach. We do not take into account, for this work, the way which the multimodal interfaces are developed.

Simon et. al. [13] investigated the authoring of multimodal interfaces on mobile devices. These authors present a prototype authoring tool that permits the development of graphical as well as multimodal web-based user interfaces. This prototype authoring tool focuses on the user that develops a service; MDIWS that focuses on the final user which seeks for a determined service.

In the Spika work [15], his concerns is about providing a multimodal software for mobile devices. Besides, it has a automatic converter dialogues to voice-based dialogues in both directions. MIDIWS can be extended for use on mobile platforms, only making a minimum adaptations to smartphones.

A limitation of VoiceXML is the use of a predetermined grammar. Rouillard [12] proposes a system of voice recognition to VoiceXML. The system has a Web service which is to recognize speech in English grammar and generates an output in English text. He uses UDDI to discovery Web services that can translate the English text to the target language. In the same sense, Hastie [3] demonstrates a DUDE (Dialogue and Understanding Development Environment). DUDE is a development environment to generate dialogue automatically, making context-sensitive Voice XML pages. MIDIWS works only with English speech. We can combine the proposal of Rouillard to make MIDIWS to work with others languages.

8 Final Remarks

In this paper, we have proposed an approach to allow the discovery and access of Web Services for users who face challenges because of device or network limitations, including small screens, restricted keyboards and lower bandwidth or for persons with impaired vision, limited motor capabilities or difficulties with perception that diminish their ability to use the Web.

Our approach is named MIDIWS – **M**ultimodal **I**nterface for **D**iscovery and **I**nvocation of **W**eb **S**ervices. In MIDIWS, a user can enter data via XHTML + VoiceXML and services will be discovered and invoked through of the Web.

We built a SOAP-Engine component in the MIDIWS architecture to enable the comunication between the Web application and UDDI. The SOAP-Engine, mediated by the Agent, deal with the messages transactions so as to make it possible to provide the search, access and return of the Web services computation.

We have implemented a case study of "Rent a Car", in order to prove that the envisaged solution is technically feasible. This case study has shown all steps of input and output of datatypes and navigation within a Web form XHTML + VoiceXML for discovery and access a service Rent a Car Company. We focus our case study on users using limited devices as devices with restricted keyboards, for instance.

We think that MIDIWS constitutes a smooth transition milestone step in the complex path towards the ultimate deployment of fully featured Web services. Our prototype can be extended to support novel uses of interaction in emerging application domains.

In future work, we plan to extend our approach to consider ontology of Semantic Web Services in the discovery the best service. In this paper we do not make any assumption regarding to the semantic of the Web service selected. We are working with straightforward discovery, based on category and service name, in UDDI [10]. We plan also to conduct a more complete and precise evaluation of MIDIWS and refine the browsing presentation strategies accordingly.

Acknowledgements

We would like to thank CAPES (Coordination for the Improvement of Higher Education Personnel), CNPq (National Counsel of Technological and Scientific Development), and RNP (National Network on Teaching and Research) for supporting the authors of this article.

References

[1] Coles, A., Deliot, E., Melamed, T., and Lansard, K. A framework for coordinated multi-modal browsing with multiple clients. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 718–726, New York, NY, USA, 2003. ACM.

[2] Dettmer, R. It's good to talk [speech technology for on-line services access]. *IEE Review*, 49(6):30–33, June 2003.

[3] Hastie, H., Liu, X., and Lemon, O. Automatic generation of information state update dialogue systems that dynamically create voice xml, as demonstrated on the iphone. In *Proceedings of the SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, SIGDIAL '09, pages 148–151, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[4] Honkala, M. and Pohja, M. Multimodal interaction with xforms. In *ICWE '06: Proceedings of the 6th international conference on Web engineering*, pages 201–208, New York, NY, USA, 2006. ACM.

[5] Lai, J. Conversation interfaces, vol. 43(9):pages.24–27. In *CACM*. ACM, 2000.

[6] Nandigam, J., Gudivada, V. N., and Kalavala, M. Semantic web services. *Comput. Small Coll.*, 21(1):50–63, 2005.

[7] Narayan, M., Williams, C., Perugini, S., and Ramakrishnan, N. Staging transformations for multimodal web interaction management. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 212–223, New York, NY, USA, 2004. ACM.

[8] Nielsen, J. *Usability Inspection Methods*, chapter Heuristic Evaluation, pages 25–62. John Wiley & Sons, New York, NY, USA, 1 edition, 1994.

[9] OASIS. Introduction to UDDI: Important Features and Functional Concepts. Available on-line on http://uddi.org/pubs/uddi-tech-wp.pdf. Visited on 07/02/2008, October 2004.

[10] Prazeres, C. V. S., Teixeira, C. A. C., and Pimentel, M. G. C. Semantic web services discovery and composition: Paths along workflows. In *Proceedings of the 2009 Seventh IEEE European Conference on Web Services*, ECOWS '09, pages 58–65, Washington, DC, USA, 2009. IEEE Computer Society.

[11] Ramakrishnan, I. V., Stent, A., and Yang, G. Hearsay: enabling audio browsing on hypertext content. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 80–89, New York, NY, USA, 2004. ACM.

[12] Rouillard, J. Web services and speech-based applications around voicexml. *Journal of Networks*, 2(1):27–35, 2007.

[13] Simon, R., Wegscheider, F., and Tolar, K. Tool-supported single authoring for device independence and multimodality. In *MobileHCI '05: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pages 91–98, New York, NY, USA, 2005. ACM.

[14] Someren, M. W. V., Barnard, Y. F., and Sandberg, J. A. C. *The Think Aloud Method: A Practical Guide to Modelling Cognitive Processes*. Academic Press, 1 edition, August 1994.

[15] Spika, M. A modular multimodal software platform for mobile devices. In *Consumer Electronics (ICCE), 2010 Digest of Technical Papers International Conference on*, pages 241 –242, jan. 2010.

[16] Sun, Z., Stent, A., and Ramakrishnan, I. V. Dialog generation for voice browsing. In *W4A: Proceedings of the 2006 international cross-disciplinary workshop on Web accessibility (W4A)*, pages 49–56, New York, NY, USA, 2006. ACM.

[17] W3C. Web Accessibility Initiative (WAI). Available on-line on http://www.w3.org/WAI/. Visited on 02/12/2008.

[18] W3C. Device independence. Available on-line on http://www.w3.org/2001/di/. Visited on 02/12/2008, 2001.

[19] W3C. Web Services Activity. Available on-line on http://www.w3.org/2002/ws/. Visited on 07/02/2008., 2002.

[20] W3C. Simple Object Access Protocol (SOAP) 1.2, W3C Recommendation. Available on-line on http://www.w3.org/TR/soap12-part0/. Visited on 07/02/2008, June 2003.

[21] W3C. Device independence, accessibility and multimodal interaction. Available on-line on http://www.w3.org/2005/04/di_mmi_wai.html. Visited on 02/12/2008, 2005.

[22] W3C. Web Services Description Language (WSDL) 2.0, W3C Candidate Recommendation. Available on-line on http://www.w3.org/TR/wsdl20/. Visited on 07/02/2008, January 2006.