Incorporating Voice Dialogs in a Multi-user Virtual Environment

Chun-Feng Liao and Tsai-Yen Li
Computer Science Department, National Chengchi University
{try,li@nccu.edu.tw}

Abstract- *The applications of 3D virtual environments and voice user interface (VUI) on personal computers has received significant attentions in recent years. Since speech is the most natural way of communication, incorporating VUI into virtual environments can greatly enhance user interaction and immersiveness. Although there have been many researches addressing the issue of integrating VUI and 3D virtual environment, most of the proposed solutions do not provide an effective mechanism for multi-user dialog management. The objective of this research is on providing a solution for VUI integration and dialog management and realizing such a mechanism in a multi-user virtual environment. We have designed a dialog scripting language called XAML-V (eXtensible Animation Markup Language -Voice Extension), based on the VoiceXML standard, to address the issues of synchronization between VUI and animation and dialog management for multi-user interaction. We have also realized such a language on a multi-user virtual environment to evaluate the effectiveness of this design.*

Keywords: Voice User Interface, VoiceXML, Dialog Management, Multi-user Virtual Environment.

1. Introduction

Due to the rapid development of graphics hardware and software, virtual reality that used to run on high-end graphics workstation can now be experienced on desktop computers. Among the potential applications, Multi-User Virtual Environment (MUVE) is one that allows many users to share their experiences in a 3D virtual environment [11]. The nature of this type of system requires tight integration of 3D graphics and distributed system technologies. An example application of this type of environment is the prevalent 3D on-line games that have received significant attentions in recent years. Other applications on military, entertainment, education, etc. are also emerging [2][15].

Most MUVE systems today, such as DIVE[4] and ActiveWorld[1], adopts a multi-model user interface consisting of 3D navigation and textual chatting. However, few of them have incorporated voice user interface, the most natural way of communication for humans, into their systems, despite the recent advances in speech-related technologies. We think the main reasons are two-fold. First, there exists no effective dialog management mechanism for multiple users across the network in general. Most of the voice applications today are simple applications focusing on the voice dialogs between a human and a machine playing the role of the other human. Second, there is no flexible way to integrate dialog specifications seamlessly into a computer-generated animation in the current MUVE systems.

In this paper, we propose a dialog management mechanism that enables the voice user interface in a multi-user virtual environment. The mechanism uses a protocol to let two avatars, representing either humans or machines, to establish a dialog connection and allow other avatars in the virtual world to observe the progress of the dialog. The protocol is an XML-based document while the dialog itself is a form based on VoiceXML [14]. Due to the extensibility of XML, this dialog management mechanism is seamlessly integrated into a MUVE system called IMNet that adopts XAML (eXensible Animation Mockup Language) [10] as the underlying animation scripting language. The voice interface is described with a language called XAML-V and embedded in an XAML script as a plug-in which can in turn trigger additional animation scripts inside the dialog.

In the next section, we will briefly review the related work in multi-user virtual environment and dialog management. In Section 3, we will describe the requirements of enabling voice dialogs in a MUVE. We will then present the design of XAML-V for realizing such a voice interface in the following section. In Section 5, we will describe some implementation issues and illustrate our design with an example dialog among multiple users. Finally, we will conclude the paper with some future research directions.

2. Related Work

2.1. Multi-user virtual environments

According to the way that a message is propagated among the users, one can roughly classify the architecture of a MUVE into two categories: *client-server* and *peer-to-peer*. In a client-server architecture, all client messages are sent to the server which in turn broadcasts the messages to all other clients[1][3]. The most common problem about this architecture is that the server can easily become a bottleneck when the number of clients

rises. However, most MUVE systems today adopt this architecture for its implementation simplicity. On the other hand, the MUVE systems adopting a peer-to-peer architecture communicate without a centralized control [5][6]. However, these systems are more difficult to implement and manage than the client-server architecture. Since system architecture is not the main concern of this work, we have chosen a MUVE system called IMNet (Intelligent Media Net)[9] with the client-server architecture.

In addition to the issue of system architecture, the application protocol for delivering multi-modal contents, such as 3D animation and textual chat, has also been an active research topic. A recent focus is on designing an XML-based animation scripting language for describing the activities in a virtual environment. For example, Avatar Markup Language (AML) [8] focuses on facial expression but only provides limited functions for altering a canned motion. STEP is another XML-based scripting language that emphasizes on its logical reasoning ability [7]. XAML is also an XML-based animation scripting language featuring its extensibility in modeling animations with various levels of controls and allowing other external modules to be incorporated as plug-in [10]. In this work, we have chosen to extend XAML to incorporate a mechanism for voice dialog management in the IMNet system.

2.2. Dialog management

The researches for core voice technologies, such as speech synthesis and recognition, and voice applications have made significant progresses in recent years. International standards such as VoiceXML are emerging as the de facto for dialog-based applications. Most of these designs aim to provide a voice user interface to a user by downloading a dialog form from a document server. However, since two-way communications between a human and a computer are usually the basic assumption for designing such a language, it cannot be directly applied to a MUVE system without modifications.

Galatea[13] is an Anthropomorphic Spoken Dialog Agent (ASDA) platform that makes use of the dialog model of VoiceXML. It extends VoiceXML to incorporate animation descriptions such as facial expression scripts. However, in [12], the authors argue that the form-filling mechanism in VoiceXML is insufficient for expressing state transitions in an advanced dialog. Therefore, a language called DialogXML is designed to express a more complex dialog. A dialog manager is also designed to translate the scripts in this language into VoiceXML scripts at run time.

3. Dialog Management in MUVE

VoiceXML was originally designed for dialogs between human and system in a telephony environment. A human user interacts with the system by retrieving a sequence of dialog forms from a document server just as we do in a typical session of a web application. In such an environment, there are at most two interactive instances in a dialog session. However, in a typical MUVE, the number of avatars in a scene is usually much larger. In a dialog session, two avatars are the active subjects while the other avatars act as observers. In order to clarify the roles of the avatars in a typical MUVE, we have adopted the following notations.

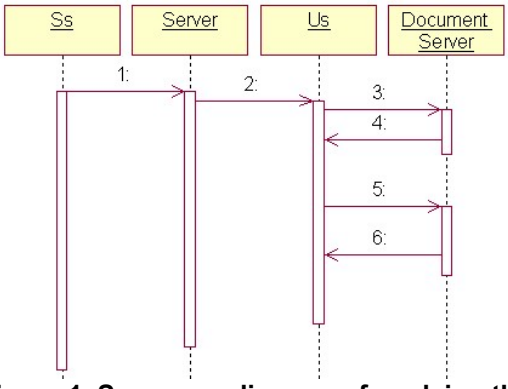

Figure 1. Sequence diagram of applying the VoiceXML dialog model to a MUVE

Subjects: Avatars in a dialog.
Observers: Avatars not in a dialog.
U: Avatars controlled by human.
S: Avatars controlled by system.
Suffix *s*: Subject avatars.
Suffix *i* (*i*=1,2,3…): Observer avatars.

For example, U_s denotes an avatar in dialog controlled by a human user.

If we adopt the dialog model of a typical VoiceXML session between two avatars controlled by a human (U_s) and a machine (S_s), the dialog may actually happen between U_s and the document server as shown in Figure 1. After the dialog is initialized, S_s sends its dialog script's URL to U_s (steps1-2), and then U_s fetches the script according to this URL from the document server, and collects inputs from the user. A new script is then fetched based on the user's response (steps 3-6).

When applying the VoiceXML dialog model to a MUVE as described above, we encountered several problems. First, although S_s is in a dialog with U_s, S_s is not aware of the dialog status after sending out the URL of the first dialog script. If some network failures occur during the dialog or U_s deliberately stops the dialog, S_s will not be notified and updated. Second, without a mechanism to maintain the dialog status, S_s may be talking to two or more avatars simultaneously or showing a mixed and confused animation to a wrong target. Therefore, we have proposed several mechanisms as described below to enhance the original dialog model.

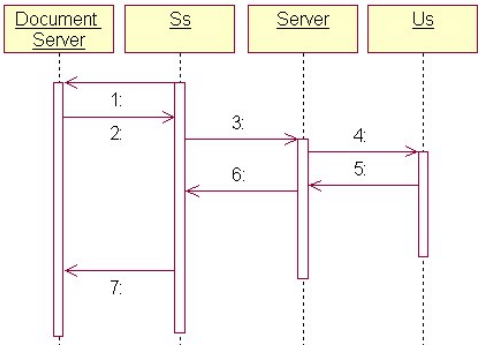

Figure 2. The sequence diagram of adopting the Proxy Request mechanism in a dialog

3.1. Proxy request

In order to make S_s be aware of the dialog status when talking with U_s, we propose to use a proxy-request mechanism as for a proxy server on WWW. In the enhanced model, all dialog requests must pass through S_s as shown in Figure 2.

With the Proxy-Request mechanism, S_s will receive all messages sent by U_s, and thus be aware of its dialog status with U_s. Therefore, S_s can detect and recover from potential errors. Since the participants of a dialog are all aware of the dialog status, the realization of many advanced dialog management mechanisms such as dialog initiation and locking as described below then become possible.

3.2. Dialog initiation and locking

Another characteristic of a dialog in a MUVE is that a user can only dedicate to a dialog session at one time. From our daily experience, we know that the output voice from one to many people is common but input voice from many people to a person is unusual. For instance, when a teacher is giving a lecture to her students, the voice is one-to-many. When many students speak out for questions at the same time, it is difficult for the teacher to understand all the questions. Therefore, we think that for a valid dialog, the output from an avatar to others may have a one-to-one or one-to-many relationship; but input from others to the avatar should only allow one-to-one relationship. To realize such a mechanism, we need to design a dialog initiation and locking process to maintain dialog states appropriately.

Before any clients can start their dialogs, they must negotiate with the other dialog partner to ensure that it is not in a dialog already. We have designed a two-round negotiation process as illustrated in Figure 3. Assume that U_s intends to have a dialog with S_s. First, U_s has to confirm that it is not in a dialog already with other clients in order to start the initiation process (step 1 in Figure 3). If this is the case, U_s will enter the "dialog negotiation" state (see Figure 4) and send a "dialog request" message to S_s (steps 2~3). If S_s is also not in a dialog,

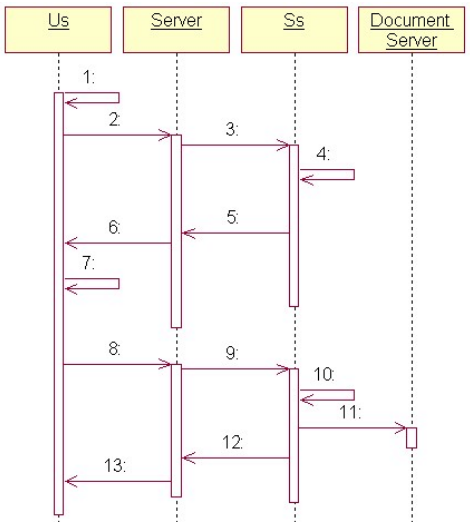

Figure 3. The sequence dialog for the dialog initiation process

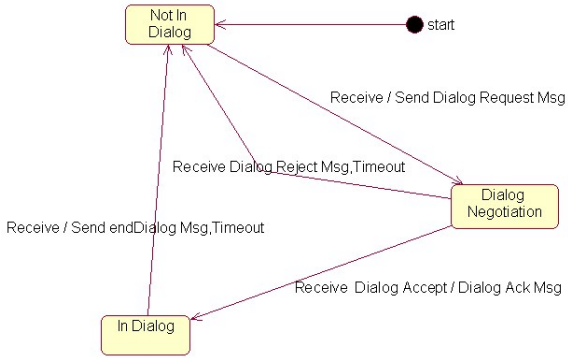

Figure 4. State diagram for dialog initiation and locking

it will enter "dialog negotiation" state as well and return a "dialog accept" message (steps 5~6) back to U_s. When U_s receives this message, it will enter the "in dialog" state and send back a "dialog accept acknowledgement" message (steps 8~9). S_s then will also enter the "in dialog" state and fetch the first dialog script from the document server for U_s (steps 10~13). On the other hand, if S_s is busy in another dialog already in step 4, it will send back a "dialog reject" message. When U_s receives this message or the process times out due to any abnormal network problems, it will enter the "not in dialog" state and abort the initiation process.

3.3. Dialog message types

The dialog initiation messages described above are sent between the two engaging parties only. However, after the dialog session starts, different avatars in a MUVE should receive different messages due to their distinguished roles in the dialog. For example, except for the engaging avatars, the other avatars are observers of the dialog. They should receive the content of the dialog but should not participate or reply to any of these dialogs. Therefore, two types of messages are designed:

```
<?xml version="1.0" encoding="UTF-8"?>
<AnimItem>
    <AnimPlugin>
        <xaml-v version="1.0">
            <block>
                <prompt>No, Thanks</prompt>
            </block>
        </xaml-v>
    <AnimPlugin>
</AnimItem>
```

Figure 5. XAML-V script as a plugin of XAML

```
<xaml-v version="1.0">
    <form id="helloForm" type="dialog">
        <prompt>Good morning
        <animation>
            <AnimItem dur="3000">
                <AnimImport src="Stand">
            </AnimItem>
        </animation>
        <field name="helpType">
            <prompt>May I help you? You can say: "I.M.
Lab", "Computer Center", or "No, thanks"</prompt>
            <animation>
                <AnimItem dur="3000">
                    <AnimImport src="Listen"/>
                </AnimItem>
            </animation>
        </field>
        <submit next="helpFormResponse.jsp" />
    </form>
</xaml-v>
```

Figure 6. Embedding animation in a XAML-V script

dialog scripts and *broadcasting scripts*. The dialog scripts are similar to a typical VoiceXML dialog form while the broadcasting scripts are like a dialog without questions. The dialog scripts are mandate and cannot be ignored while the broadcasting scripts may be safely ignored by other avatars if necessary.

4. Design of XAML-V

A scripting language called XAML-V (XAML Voice extension), an extension of XAML, is designed to realize the voice user interface and dialog management described in the previous section. In this section, we will describe the scripting language in more details to illustrate how it takes advantage of the extensibility of XAML to make the animation scripting language speech-enabled. XAML-V mainly consists of tags with two types of functions: *dialog context and dialog management protocol*.

4.1. Dialog context

XAML is an animation scripting language that allows other modules, such as XAML-V, to be incorporated as plug-ins. As shown in Figure 5, a XAML-V script is enclosed in the <xaml-v> tag, which is embedded in an <AnimPlugin> tag. The dialog context part of XAML-V is based on a subset of VoiceXML with the telephony-related elements removed since they are not appropriate in MUVE.

```
<xaml-v>
    <protocol>
        <dialog-negotiate source="Us" context="request"/>
    </protocol>
</xaml-v>
```

Figure 7. Dialog request message

```
<proxy-request>
    <method>GET</method>
    <url>helloFormResponse.jsp</url>
    <parameter>
        <param key="helpType" value="no thanks"/>
    </parameter>
</proxy-request>
```

Figure 8. An XAML-V script for proxy request

For example, the tags of <block>, <prompt>, <form>, and <field> all bear the same meanings as they are in VoiceXML while <transfer>, <filled>, and <assign> are removed.

In addition to the VoiceXML-related tags, XAML-V also supports embedded animations inside a dialog at both the form level and the field level. The embedded animations are XAML scripts that do not recursively include XAML-V scripts. For example, in Figure 6, a form-level and a field-level animation that imports canned motions from external files through the <AnimImport> tag is used.

The XAML-V script example in Figure 6 describes a scenario where a computer-controlled avatar welcomes the user by a greeting statement "Good Morning, sir. May I help you?" Then the system asks the user where he/she is interested in going while playing a high-level "listen" animation clip at the same time to prompt the user for a response. The response will then be sent to the given URL for further processing.

4.2. Dialog management protocol

Several tags are added to support the dialog management mechanism proposed in the previous section. Figure 7 shows an example of dialog negotiation message. The "context" attribute indicates the type of dialog negotiation being executed, and the "source" attribute indicates where this message is from.

In Figure 7, the "context" attribute is "request", and the "source" is "Us". The script means that an avatar "Us" would like to "request" a conversion with the user. The possible values for the "context" attribute of the dialog-negotiation element include: *request*, *accept*, *reject*, *dialogAck*, and *endDialog*. Each of these values maps to an action in a dialog negotiation process described in the previous section.

Figure 8 shows an example of the Proxy Request mechanism in XAML-V. The idea is to encapsulate HTTP GET/POST messages in the <proxy-request> tag such that the system-controlled avatar can fetch the next document from the document server. In the <proxy-request> element, the HTTP method,

Table 1. Comparison of implementation in various MUVE's with voice user interface

System	Cernak[4]	Wauchope MSFT[15]	Wauchope ISFS[16]	XAML-V
Virtual environment	VRAC's C6	EA's World Toolkit	Cortona VRML Browser	IM-Browser
Speech recognition	CSLU Toolkit	IBM ViaVoice 8	IBM ViaVoice 8	IBM ViaVoice 9
SR grammar	Home made	IBM SRCL	JSGF	SRGF (W3C Standard)
SR invocation	Keyword	Not mentioned	Push to talk	Push to talk
TTS	Festival	IBM ViaVoice 8	IBM ViaVoice 8	IBM ViaVoice 9
Speech API	Not mentioned	IBM SMAPI	JSAPI	JSAPI w/ Cloud Garden Bridge
Speech-VR bridge	TCP Socket	TCP Socket	UCP Socket	TCP Socket
Dialog flow control	SCI IDE	Rule and data stored in RDBMS	Rule and data stored in RDBMS	XAML-V

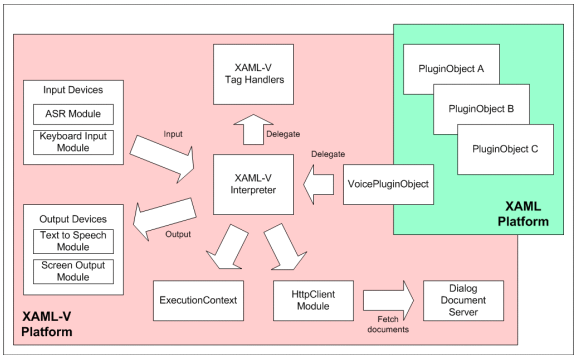

Figure 9. System architecture of XAML-V platform

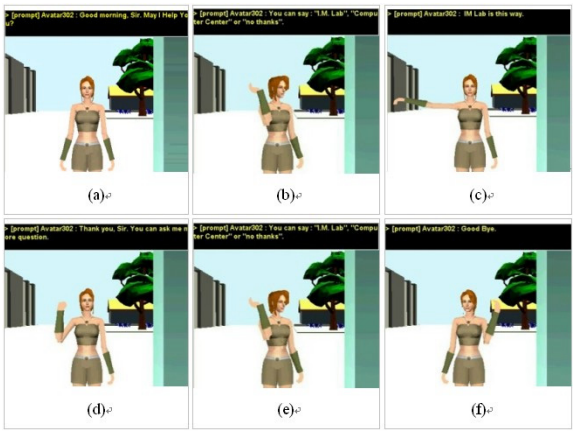

Figure 10. Snapshots of the interface for an example dialog in a MUVE

requesting URL, and requesting parameters are the sub-elements to encapsulate detail information.

5. Implementation and Example

5.1. Implementation

We have implemented the enhanced dialog model with XAML-V in IMNet [9][10]. The XAML-V module serves as a plug-in component of the XAML platform and coordinates with various input and output devices. The XAML-V module interprets XAML-V script ands manages several dialog mechanisms (e.g. dialog lock or dialog state). A comparison of the implementation of XAML with other speech-enabled MUVE systems is summarized in Table 1.

Figure 9 shows the overall architecture of XAML-V platform. The *VoicePluginObject* serves as a plug-in point to XAML platform. It accepts scripts from the XAML platform and delegate to a XAML-V interpreter. The *XAML-V interpreter* is the core of the XAML-V platform, which parses incoming scripts and orchestrates the other components. *ExecutionContext* is the data store for run-time configurations and information needed by the interpreter. The *dialog document server* is a repository for dialog scripts. These scripts may also be generated dynamically using server-side scripting technologies. For example, we use an open source Java Servlet container (Tomcat 4.1) as the dialog document server in our implementation. The *HttpClient* fetches dialog scripts from the document server and handle HTTP protocol details for the interpreter. *Tag Handlers* are collections of classes conformed to a "TagHandler" interface, and each of them is designed to handle a specific tag. The interpreter delegates work to this component according to the tags that it encounters. For example, it will delegate work to PromptTagHandler class if the interpreter encounters a <prompt> tag. In addition to rendering the voice with the TTS module, the PromptTagHandler object will send out a broadcasting message containing a <prompt> script to let all other avatars render the voice as observers.

According to the plug-in model of XAML, when the interpreter encounters the <AnimPlugin> element, it will search a pre-configured component registry for a valid plug-in to handle the script described inside the <AnimPlugin> element. The XAML interpreter will acquire the control of current executing thread and delegate to a plug-in component when it finds one. Since XAML-V is actually the plug-in component, the XAML-V interpreter will take over the control of current thread and continue to execute the script.

5.2. An Example

In Figure 10, we show the snapshots of the user interface for an example of interactive animation with a voice dialog written in XAML and XAML-V. The example dialog script is similar to the one shown in Figure 6. In the scenario, a virtual

Figure 11. Snapshots (b and c) of the above dialog from an observer's viewpoint

character acts as a receptionist via the voice user interface when a real user enters the virtual environment. The receptionist greets the guest by saying "Good morning, Sir, May I help you?" (Figure 10(a)) Then he will listen to the user's input for the destination that he/she is interested in and play a high-level animation "listening" at the same time (Figure 10(b)). If the user does not need any assistance, the receptionist will end the dialog by saying "Good-Bye" (Figure 10(f)). If the user specifies one of the destinations that the receptionist knows, she will guide the user to the destination (Figure 10(c)). Unless the user says "No, thanks", the receptionist will continue to ask the user for further question (Figures 10(d) and (e)).

Figure 11 shows two snapshots (corresponding to (b) and (c) in Figure 10) of the dialog from the observer's view. The avatar with blonde hair is the observer of this dialog. She can hear all speech voices of the dialog, or may choose to ignore these voices safely if she would like to have a dialog with another avatar.

6. Conclusions and Future Work

In this paper, we have proposed to enhance MUVE with a voice user interface. We have presented a dialog management mechanism for MUVE based on VoiceXML and XAML. The proposed XAML-V dialog scripting language includes functions on dialog lock, dialog broadcasting, dialog negotiation, and a proxy request mechanism. We have demonstrated the appropriateness of this design by examples and shown that by integrating with an appropriate voice interface, users can communicate with each other in a more natural way in MUVE.

We have been focusing on realizing the dialog management mechanism for MUVE; however, many desirable features still need to be added to enhance the immersion of the virtual environment. For example, the volume of the voice dialog as well as other 3D sound effects should be adjustable according to the relative locations between avatars. In addition, a more attractive facial animation synchronized with the voice dialog should be adopted to enhance visual realism.

This work was partially supported by a grant from National Science Council under contact NSC 93-2213-E-004-001.

References

[1] ActiveWorlds, URL:<http://www.activeworlds.com>.

[2] O. Apaydin. "Networked Humanoid Animation Driven by Human Voice using Extensible 3D (X3D), H-Anim and Java Speech Open Standards," *Master's Thesis*, Naval Postgraduate School, March 2002.

[3] Blaxxun, URL:<http://www.blaxxun.com>

[4] M. Cernak and A. Sannier, "Command Speech Interface to Virtual Reality Applications," Technical Report, Virtual Reality Applications Center at Iowa State University of Science and Technology, June 2002.

[5] E. Frecon and M. Stenius, "DIVE: A Scalable network architecture for distributed virtual environments," *Distributed Systems Engineering Journal* (Special issue on Distributed Virtual Environments), Vol. 5, No. 3, pp.91-100, September 1998.

[6] C. Greenhalgh and S. Benford, "MASSIVE: a collaborative virtual environment for teleconferencing," *ACM Trans. CHI, Vol.2*, pp.239-261, 1995.

[7] Z. Huang, A. Eliens, and C. Visser, "STEP: A Scripting Language for Embodied Agents," *Proceedings of the Workshop on Lifelike Animated Agents*, 2002.

[8] S. Kshirsagar, A. Guye-Vuilleme, and K. Kamyab, "Avatar Markup Language," *Proc. of 8th Eurographics Workshop on Virtual Environments*, pp. 169-177, May, 2002.

[9] M.Y Liao, "An Extensible Scripting Language for Interactive Animation," Master's thesis, Department of Computer Science, National Chengchi University, 2004.

[10] M.Y Liao and T.Y Li, "A Scripting Language for Extensible Animation," *Proc. of 2003 Computer Graphics Workshop*, Taiwan, 2003.

[11] M. Matijasevic, "A Review of Networked Multi-User Virtual Environment," URL: <http://citeseer.nj.nec.com/matijasevic97review.html>, 1997

[12] E. Nyberg, T. Mitamura, P. Placeway, M. Duggan, and N. Hataoka, "DialogXML: Extending VoiceXML for Dynamic Dialog Management," *Proc. of the Human Language Technology Conf.*, 2002.

[13] S. Sagayama, S. Kawamoto, H. Shimodaira, T. Nitta, T. Nishimoto, S. Nakamura, K. Itou, S. Morishima, T. Yotsukura, A.Kai, A.Lee, Y. Yamashita, T. Kobayashi, K. Tokuda, K. Hirose, N. Moinematsu, A. Yamada, Y. Den, and T. Utsuro, "Galatea:An Anthropomorphic Spoken Dialogue Agent Toolkit," *IPSJ SIG-SLP*, Feburary 2003

[14] VoiceXML, URL:< http://www.w3.org/Voice/>

[15] K. Wauchope, "Interactive Ship Familiarization System: Technical Description," AIC Technical Report AIC-03-001, Navy Center for Applied Research in Artificial Intelligence, Washington DC, 2003.

[16] K. Wauchope, S. Everett D. Tate, and T. Maney, "Speech-Interactive Virtual Environments for Ship Familiarization," *Proc. of 2nd Intl. EuroConference on Computer and IT Applications in the Maritime Industries (COMPIT '03)*, pp. 70-83, Hamburg, Germany, May 2003.