Zanzibar OpenIVR: an Open-Source Framework for Development of Spoken Dialog Systems

Dmytro Prylipko[1], Dirk Schnelle-Walka[2], Spencer Lord[3], and Andreas Wendemuth[1]

[1] Chair of Cognitive Systems, Otto-von-Guericke University Magdeburg
Magdeburg, Germany
{dmytro.prylipko, andreas.wendemuth}@ovgu.de
[2] Telecooperation Lab, Darmstadt University of Technology
Darmstadt, Germany
dirk@tk.informatik.tu-darmstadt.de
[3] Spokentech, Inc., San Francisco, CA
spencer.lord@gmail.com

Abstract. The maturity of standards and the availability of open source components for all levels of the MRCP stack provide us with new opportunities for the development of spoken dialog technology. In this paper a standard-based and modular architecture for interactive voice response (IVR) systems is presented together with its implementation – Zanzibar OpenIVR. The architecture, described in terms of components and standards, is compared to other existing frameworks. The usage of our framework is discussed regarding different aspects of spoken dialog technology such as speech recognition and synthesis, integration of the components, dialog management, natural language understanding. It is designed to work over VoIP as well as with usual telephony communication channels, thus provides an ability for web based access. Zanzibar OpenIVR is able to serve as a starting point for building dialog systems and research in voice-enabled technologies.

Keywords: IVR, VoiceXML, VoIP, Spoken dialog system

1 Introduction

Since Bolt's *Put that there* [2] speech is considered to be among the most important means of post desktop interaction. Considerable progress in the development of voice based interfaces has been achieved during the recent years, bringing us to natural access to information.

Following the W3C voice browser activity, state-of-the-art spoken dialog systems provide functions such as call handling, speech recognition and synthesis, retrieval of data and dialog management together with language understanding. Looking ahead, modern systems should provide access via VoIP as well as existing communication lines (GSM and PSTN). IP telephony is more flexible than regular PSTN lines and provides us with ample opportunities for call routing

and management. Moreover, it allows integration with services available over the Internet, e.g. with click-to-call technology, thus aiming at what is generally called the *voice-enabled web*.

Recently, several proprietary frameworks have been developed for building spoken dialog systems. Whereas prices are quite reasonable, we adhere to an opinion that open source solutions not only save money, but also encourage developers and scientists to study and improve technology. Open standards for a particular field as well as the corresponding software, thereby offer a playground for modifications and full control during the development stage.

A recent study [11] showed that there are also serious security issues using third party software. The conjunction with the findings of Hammonds et al. from Forrester Research [4] who claim that the average bug fixing time for enterprise developers is 6.9 days while 36% of the open source developers need under 8 hours to fix a bug since it was discovered, shows the potential of open source vs. proprietary software.

Although a significant number of open source application frameworks have been designed in the recent years, these solutions are often in-house tools, tied to particular products or custom APIs, which leads to huge efforts for learning and using such systems.

That is why special attention should be given to resort to open standards and pluggable components in order to make the whole framework transparent, flexible and reusable. Additionally, these standards should be suitable to be used in research projects by enabling to rely on functional building blocks while keeping the opportunity to modify the default behavior, to go beyond the state of the art.

In this paper we describe such a modular open source framework for spoken dialog systems that meets these special requirements within research and development. Thereby, we address the need for a common test-bed. We also provide a small overview of existing systems.

2 The Architecture of Zanzibar OpenIVR

Zanzibar OpenIVR[4] uses the JVoiceXML interpreter and contains a MRCPv2 Server with the Sphinx4 speech recognizer and FreeTTS speech synthesizer. It integrates with a VoIP PBX (Private Branch Exchange) (like Asterisk) using SIP and RTP VoIP standard. It can deploy and run VoiceXML documents as well as applications written in Java.

An overview of the Zanzibar's architecture is presented in Fig. 1. A more detailed description of component interactions is given in section 2.2.

2.1 Components

Telephony Server. The telephony subsystem functions as the gateway to the voice network. It provides access to various telephony protocols and networks.

[4] `http://sourceforge.net/projects/openivr/`

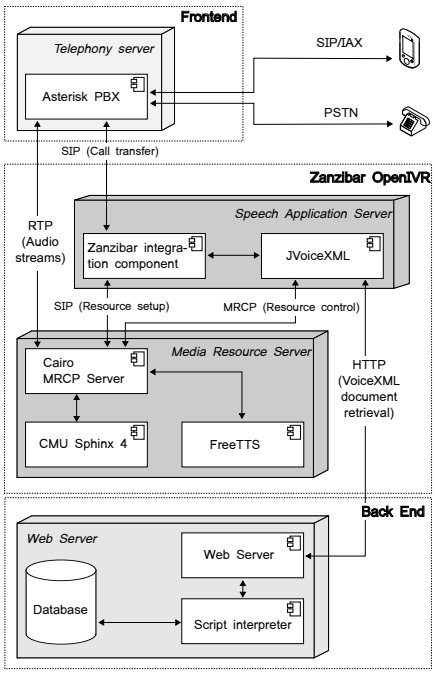

Fig. 1. Architecture of a spoken dialog system built on Zanzibar OpenIVR

It does routing and call handling for incoming calls, provides flexible dialplan management. It must be a robust and scalable server and should have a rich API and extension framework that provides the ability to add modules that can inetgrate with other components. It should be based on open source solution for the advantages over proprietary solutions mentioned above.

Our proposed solution employs Asterisk[5] as a telephony interface because of its popularity, continuous development and reliable support.

Speech Application Server. After the call has been received by the telephony server, the processing of the dialog starts. The small survey of established frameworks in section 3 shows that VoiceXML is suitable to fulfill our request for a standardized dialog control component. As a domain specific language, VoiceXML allows for task independent, modular and reusable dialog development. The major advantages of using VoiceXML are:

- It is an open W3C standard independent of any existing implementation (it has the status of W3C recommendations since 2000). It is supported by a set of powerful tools for dialog development;
- VoiceXML applications are less expensive in development compared to traditional IVRs [5].

[5] `http://www.asterisk.org/`

– It provides mechanisms for dynamic management of the spoken dialog.

JVoiceXML[6] is an open source VoiceXML browser written entirely in the Java programming language, supporting the VoiceXML 2.1 standard.

Besides the support of Java APIs such as JSAPI and JTAPI, custom speech engines can easily be integrated into this platform. Examples are a text based platform and an MRCPv2 platform that are shipped with the voice browser.

Media Resource Server. The Media Resource Control Protocol (MRCP) is a communication protocol designed to provide a mechanism to control processing resources on the network by a client. These resources are usually speech recognizers (ASR engines) and speech synthesizers (TTS engines). MRCP allows the implementation of distributed interactive voice systems, for instance VoiceXML interpreters. In such a system, the voice browser acts as an MRCP client, while the MRCP server provides an access to ASR and TTS engines. It has additional functionality such as load balancing, clustering and failure handling in order to meet the scalability requirements. The key benefit in the separation of the VoiceXML interpreter and the media resource server is the independence of components and their ability to easily be replaced.

Zanzibar OpenIVR uses Cairo[7] as a media resource server. It implements a subset of the MRCP version 2 specification and integrates with CMU Sphinx 4 and FreeTTS which are described below.

Speech Recognition Engine. For speech recognition the CMU Sphinx 4 [12] is employed. The recognizer is a state-of-the-art continuous-speech, speaker-independent systems based on hidden Markov models (HMMs) and N-gram statistical language models.

CMU Sphinx 4 provides numerous capabilities: generalized pluggable frontend and language model architectures, rich capabilities for language modeling, generalized acoustic model architecture, utilities for post-processing recognition results (obtaining confidence scores, generating lattices), speaker adaptation mechanisms. CMU Sphinx is distributed under an academic BSD-style license. The code and binaries are free for commercial and non-commercial use.

Text-to-Speech Engine. FreeTTS[8] is an open source speech synthesis system based upon CMU's Flite, which is by-turn derived from the Festival and the FestVox project, from Carnegie Mellon University. FreeTTS is released under BSD license.

FreeTTS supports a number of voices (including MBROLA voices) and is also able to import vocies from FestVox. It implements the speech synthesis part of JSAPI 1.0.

[6] `http://www.jvoicexml.org/`
[7] `http://www.speechforge.org/projects/cairo/`
[8] `http://freetts.sourceforge.net`

2.2 Integration Issues

Figure 1 shows how all the components integrate together in the Zanzibar OpenIVR to form a coherent system. Zanzibar consists of three servers: the teltphony server, the speech application server and the media resource server.

The PBX server acts as a gateway between the users and Zanzibar. When the call is to be routed to the IVR, the transfer utilizes the SIP protocol by the `Dial` command from the Asterisk dialplan to connect the telephony server with the speech application server. The use of the SIP protocol enables the use of any other standard PBX.

The speech application server interacts with the telephony platform, establishes, maintains and tears down sessions, runs the application for the user and gathers speech resources as needed from the MRCPv2 server.

VoiceXML functionality is enabled by the use of JVoiceXML running as an embedded component that is hooked via the Zanzibar integration component. Speech recognition and synthesis are not provided by JVoiceXML itself, but by the external systems, so-called *implementation platforms*. JVoiceXML has a framework to add different implementation platforms. The primary mechanism to do this is through a system out- and user input Service Provider Interface (SPI). Such an implementation has been done in the MRCP4J client library, which is used to integrate JVoiceXML with its implementation platform - Cairo MRCP server.

As one can see from the Fig. 1, the audio streams go directly from the media resource server to the telephony server bypassing the speech application server. The audio is transferred to the PBX and back via RTP protocol. CairoRTP library over Java Media Framework (JMF) is used for that.

2.3 Dialog Management and Natural Language Understanding

An analysis in [9] names three dialog strategies that can be used in the development of voice user interfaces: user initiative, system initiative and mixed initiative.

VoiceXML claims to support all three, although not all are explicit goals. The most common dialog strategy that is used in current telephony applications is system initiative. User initiative dialogs are rarely used, although they are possible to develop. VoiceXML is designed around the concept of *form items* that are divided into *input items* where the user can enter some data and *control items* containing executable code. Both input and control form items are designed to support the development of system initiative dialogs.

In a mixed initiative dialogs the user can take the initiative. If the system can not fulfill the request from the user because the request is missing some data, the system can take the initiative to ask for the missing data. In this case the system essentially falls back to form filling mode. This comes closer to the vision of a human-like conversation. In this case the concepts of VoiceXML are still valid. The user's input is still restricted to match certain grammars. Hence, information overloading is not possible. Also, there is no real support for intention recognition

to determine the user's goal for calling. It remains restricted to the current dialog context. Summing up, the way VoiceXML treats mixed-initiative is not what is really mixed-initiative but slot-filling.

With Zanzibar we are also limited to the concepts of VoiceXML. However, many applications based on this technology have been successfully developed and deployed. Efficient dialogs, coming close to a natural interaction, are only possible with carefully designed dialog scripts. Many guidelines tell how to achieve that, like the one given in [3] exist. There are also first attempts to transfer the idea of design patterns [9] to the design of voice user interfaces.

3 Related Work

In the recent past, several solutions for spoken dialog have been presented. The Thai voice application gateway [6] is similar to the framework proposed in this paper. Unfortunately, this branch has not been ported back to the origin JVoiceXML source code and the entire system is not available anymore because of technical reasons.

Olympus [1] is a framework for spoken dialog system created at Carnegie Mellon University (CMU). At the high level it consists of a series of components connected in a pipeline architecture. Several spoken dialog systems have been successfully developed and deployed using Olympus framework (Let's Go!, LARRI, TeamTalk etc.).

The Jaspis open framework [10] provides a radical change in the way how spoken dialog systems are developed, moving away from frame-based application design. Hence, it establishes its own proprietary standard with a limited user group, while we rely on established standards to address a broader community.

Noteworthy solutions of commercial software are the Sympalog speech technology [7], InterpreXer VoiceXML IVR Server[9], OptimTalk[10] and framework built with Asterisk + VXI* VoiceXML browser[11] together with supported ASR and TTS engines. A framework presented in [8] also provides a platform for development of multi-purpose dialog systems, but is built with some commercial software.

Only four of the frameworks mentioned above (Thai voice application gateway, Voxy over Asterisk, OptimTalk and VXI*) are based on a standardized programming interface for dialog control and modeling which is VoiceXML in all cases.

Whereas one can see that number of commercial as well as research solutions for spoken dialog systems is quite significant, we have to state that no full open-source stack based on VoiceXML 2.1 standard is available.

[9] `http://www.phonologies.com/interprexer.php`
[10] `http://www.optimsys.cz/technology/introduction.php`
[11] `http://www.i6net.com/products/vxi/`

4 Evaluation and Discussion

The work reported in this paper results from an integration of existing open source components with open standards into the particular framework, namely Zanzibar OpenIVR. The cooperation with several research related institutes prove Zanzibar to be a good starting point for the development of voice based applications.

In Table 1 a comparison of several currently available frameworks is presented from the point of view of flexibility, openness and integration capabilities. Under

Table 1. Comparison of the frameworks for spoken dialog systems

Framework	Integration with IP-PBX	Modular	VoiceXML based	Open source
InterpreXer VXML Server	+	+	+	–
Jaspis	+	+/–	–	+
Olympus	–	+/–	–	+
OptimTalk	+	+	+	–
Sympalog	+	+	–	–
VXI* VoiceXML Browser	+	+	+	–
Zanzibar OpenIVR	+	+	+	+

integration with IP-PBX we mean the ability of the framework to deal with the telephony system via standard protocol like SIP or IAX. The system is considered to be modular if it allows to interchange components from different vendors, which can be achieved, for instance, using standards. For instance, Jaspis and Olympus are open, modular and transparent, however based on in-house infrastructure which hampers integration.

The key feature of the described framework is its flexible nature: It relies on standards developed for the spoken dialog technology and does not depend on a particular implementation therefore. In its current configuration (Asterisk, JVoiceXML, Cairo, Sphinx, FreeTTS) it presents one possible combination of components. Other ones can be created as needed.

All of the related frameworks evaluated in this paper can be used to evaluate things like dialog strategies, acoustic and language models within that particular framework. But the standards-based, modular nature of the Zanzibar design provides an additional advantage over the other related frameworks. It can also be used for comparing various components, e.g. speech recognition engines or VoiceXML interpreters on the same task.

It exposes standards and provides a testbed for their development by common effort. Commercial software also provides such a possibility, but often in a limited amount: a support for a non-standard component almost always requires a corresponding update by request.

5 Conclusion and Further Work

In this paper we described Zanzibar OpenIVR – an open source framework for the development of telephone-based voice-enabled applications. It can be useful in research and development of communication standards, components of conversational systems, testing dialog flows and acoustic models etc. However some shortcomings remain, mostly related to technical imperfections. In its future development we will try to make it faster, more reliable and easier in installation and configuration. These goals can be accomplished with further improvements like pooling of VoiceXML sessions to decrease the response time or less complicated installation procedure with auto-registration in Asterisk.

We hope that the overview of frameworks for conversational systems given in this paper will make the landscape of proposed solutions less daunting, and help researchers choose the most appropriate platform for a given task.

References

1. D. Bohus, A. Raux, T. K. Harris, M. Eskenazi, and A. I. Rudnicky. Olympus: an open-source framework for conversational spoken language interface research. In *NAACL-HLT '07: Proceedings of the Workshop on Bridging the Gap*, pages 32–39, Morristown, NJ, USA, 2007. Association for Computational Linguistics.
2. R. A. Bolt. Put-that-there: Voice and gesture at the graphics interface. *SIGGRAPH Comput. Graph.*, 14:262–270, July 1980.
3. M. H. Cohen, J. P. Giangola, and J. Balogh. *Voice User Interface Design*. Addison-Wesley, Boston, Jan. 2004.
4. J. S. Hammond, M. Gerush, and J. Sileikis. Open source software goes mainstream. Research document, Forrester Research, Apr. 2009.
5. E. Jackson. Speaking up for cost savings in the call center: Vxml takes on the dinosaur of legacy ivr. `http://www.thefreelibrary.com/Speaking+up+for+cost+savings+in+the+call+center:+VXML+takes+on+the...-a0107216561`, Aug. 2003. last accessed 08/20/2010.
6. D. Kaitrungrit and M. N. Dailey. Thai voice application gateway. In *Proceedings of ECTI-CON'08*, pages 101–104. Ieee, May 2008.
7. E. Nöth, A. Horndasch, F. Gallwitz, and J. Haas. Experiences with Commercial Telephone-based Dialogue Systems (Erfahrungen mit kommerziellen Telefon-Sprachdialogsystemen). *it - Information Technology*, 46(6):315–321, June 2004.
8. J. N. Nuno, J. P. Neto, N. J. Mamede, R. Cassaca, and L. C. Oliveira. The Development Of A Multi-Purpose Spoken Dialogue System. In *Proceedings of EUROSPEECH*, 2003.
9. D. Schnelle. *Context Aware Voice User Interfaces for Workflow Support*. PhD thesis, TU Darmstadt, 2007.
10. M. Turunen and J. Hakulinen. Jaspis – a framework for multilingual adaptive speech applications. In *Proceedings of the 6th International Conference on Spoken Language Processing, Beijing*, 2000.
11. Veracode. State of software security report volume 2. Research report, Veracode, Sept. 2010.
12. W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh, E. Gouvea, P. Wolf, and J. Woelfel. Sphinx-4: a flexible open source framework for speech recognition. Technical report, Mountain View, CA, USA, 2004.