A DISTRIBUTED MULTIMODAL DIALOGUE SYSTEM BASED
ON DIALOGUE SYSTEM AND WEB CONVERGENCE

Feng Liu, Antoine Saad, Li Li, Wu Chou
Avaya Labs Research
233 Mountain Airy Road
Basking Ridge, New Jersey 07920, USA
(fengliu,agsaad,lli5,wuchou)@research.avayalabs.com

ABSTRACT

We describe a distributed multimodal dialogue system architecture based on the concept of hybrid-VoiceXML. It utilizes a special hybrid-construct to integrate multiple multimedia, multimodal processes into one dialogue that includes VoiceXML as its voice modality. The hybrid-construct in our approach has several important functions. It provides an additional abstraction layer for dynamic dialogue generation, which greatly improves the efficiency and flexibility of the dialogue system. Under the proposed approach, the dialogue control between each interaction channel can be exchanged through the interface of a dynamic XML page. Several case studies are performed. It indicates that the proposed hybrid-VoiceXML approach is highly extensible. It can be used to form platform independent and distributed extensions for multimodal dialogue interaction beyond voice.

1. INTRODUCTION

The advent of powerful communication devices has made multimodal interaction beyond voice a reality. Numerous applications have been developed through PDA, IP terminals and 2G-3G cell phones, etc. The rapid deployment of multimodal applications has a profound impact on the design and implementation of a dialogue system that is traditionally intended for interaction with voice. One of the assumptions made in voice dialogue system is that the communication between the user and the system is through a telephone line following a single modality communication model where events are happening time sequentially as in a steam line synchronized process.

This assumption is no longer valid in a multimodal interaction based dialogue system. The complexity of multimodal interaction can be depicted in a four level hierarchy based on the approach (with revision and addition) proposed by Niklfeld et. al. [5].

Communication mode in multimodal interaction:

1) Sequential multimodal interaction: Either one or the other input and output modality is active, but never more than one simultaneously.

2) Uncoordinated, simultaneous multimodal interaction: It allows concurrent activation of more than one modality. However, should input be provided by more than one modality, these information are not integrated but will be processed in isolation, in random or specified order.

3) Coordinated, simultaneous multimodal interaction: It allows concurrent activation of more than one modality for integration and complementary into joint events based on time stamping.

4) Collaborative, information overlay based multimodal interaction: In addition to (3), it utilizes a shared common multimodal information environment (white board, web form, game console, etc.) for multimodal collaboration with information overlay.

Each level up in this hierarchy represents a new challenge for dialogue system design and departs farther away from the single modality communication by voice model.

In this paper, we describe an approach of a distributed multimodal interaction dialogue system based on the concept of dialogue system and web convergence, utilizing a web based dialogue markup language. A new architecture Hybrid-VoiceXML is proposed. It allows the combination of multiple multimedia, multimodal processes in one dialogue. Dialogue control in the proposed approach can be exchanged between different processes through the interface of dynamic dialogue page. We introduce a server page based abstraction layer for dynamic multimodal dialogue interaction. The proposed approach is applied successfully to several multimodal interaction applications for customer relation management utilizing distributed multimodal resources.

2. DIALOGUE SYSTEM AND WEB CONVERGENCE

The ubiquitous of web leads to a major paradigm shift in dialogue system design. VoiceXML is a standard effort of W3C to use voice to enable web applications. It is based on a web language from XML (Extensible Markup Language). It provides a level of dialogue abstraction that separates the dialogue design from the details of the physical resources.
Applications developed using VoiceXML is highly portable across various platforms. From dialogue system architecture point view, VoiceXML is an application tier infrastructure, and it is based on a convergence between voice dialogue system and web. It distinguishes from a legacy interactive voice response system in many ways, as tabulated below.

1. Interpretative and no compiling process in which dialogue action can be made dynamic and fully interpretative based on the current history of the dialogue interaction.
2. A web based knowledge environment in which the huge knowledge base of web can be utilized for constructing dialogue actions.
3. Integration of HTTP and web interface to dialogue system, in which an environment for thin-client and service hosting is created.
4. HTTP based service transaction that is beyond the traditional voice and telephone interface.

The web convergence implemented in VoiceXML has a profound impact on the architecture of voice dialogue systems. However, for multimodal interaction, new approaches are needed and the single procedure telephone/dialogue system communication model for interaction by voice is no longer fit. One issue is how to integrate multi-process, multimodal communication in a distributed dialogue system through the convergence with web and leveraging the success of VoiceXML as its voice modality.

3. Hybrid-VoiceXML FOR MULTIMODAL INTERACTION

As characterized in the four level hierarchies, multimodal interaction is intrinsically a multi-input, multi-procedure interaction process. The dialogue system in our study follows a classical three-tier client-server architecture [1]. The first layer of the system is the physical resource tier (e.g. IP terminal, telephone server, email server, ASR, TTS, etc.). The second layer of the system is the API tier. It wraps all physical resources of the first tier as APIs. These APIs are exposed to the top-level application tier for dialogue applications. In order to maintain the separation of the service logic and the low level physical resources, we focus on a distributed application tier infrastructure that can support multimodal interaction and, at the same time, has VoiceXML as its voice modality.

Although VoiceXML allows certain customer extensions through the use of special <object> tag, they are mainly a vertical extension to expose new platform APIs to the interpreter. Such object extension is tied to the platform and quite restrictive in many ways. To support distributed multimode interaction, new infrastructure at the application tier is needed. It should provide an extensible and flexible environment for application development so that new issues, current and potentially the future ones, can have a chance to be addressed without a complete overhaul of the whole platform. Most importantly, the application tier infrastructure should be sharable cross multiple platforms with re-usable and distributed components. This will provide maximum flexibilities in forming joint event in multimode interaction.

As a solution to these critical issues, a hybrid infrastructure, hybrid-VoiceXML is developed for multimodal interaction beyond voice. A block diagram of hybrid-VoiceXML is depicted in Fig. 1. It is based on a distributed structure, hybrid-construct, to integrate multiple modalities in one joint dialogue action and using VoiceXML as its voice modality. In hybrid-VoiceXML, it allows the application tier to be distributed over multiple platforms, or acting as a distributed component for multiple multimodal interactions, all through the HTTP interface. Multiple processes for different modalities can be launched at the hybrid-construct. The communication between the hybrid-construct and VoiceXML interpreter is through VoiceXML pages. From VoiceXML interpreter side, it takes the dynamically generated pages from the hybrid-construct to perform interaction through the voice channel. The interaction and request from the voice channel can be submitted through the VoiceXML interpreter to the hybrid-construct and used by other interaction modalities. The dialogue control can be exchanged between the VoiceXML interpreter and the hybrid-construct, depending on who issues new VoiceXML page and who currently holds the dialogue handle. This is important, since in multimodal interaction, dialogue interaction can happen at any individual modality channel, in addition to happen jointly.

![Figure 1: A distributed hybrid-VoiceXML](image)

Fig 2 is a more detailed view of the hybrid-construct in our proposed approach. It has a special dialogue agent (DA) layer. Each DA in the DA layer has one end that connects to the server page behind web server through a message queue and another end to the interaction multimodel server. The interaction is managed through the DA and message queue combination. One of the important features of the hybrid-construct in hybrid-VoiceXML is that it can be exposed as a distributed multimodal interaction resource and is not tied to the platform. Once it is constructed, it can be hosted and share used by different process or even different platforms. This is fundamentally different from the <object> tag extension in VoiceXML.

The DA in the hybrid-VoiceXML servers several purposes. In our approach, it is a dialogue abstraction and management layer for dynamic dialogue generation. For example, when a requirement is submitted from VoiceXML to the hybrid-construct, the server page is activated and packs the requirements into messages and sends them to all message queues, which connect to DA. Each DA in this case can be realized with a COM object. When DA receives the message
from the message queue, it unpacks the message, translates the requirement into the proper format and sends it to the server. After receiving the interaction result from that server, DA converts the result into a VoiceXML page and sends it back to the server page through the receiving message queue with channel ID information included. Server page keeps and synchronizes all information with the result from DAs. It will properly pack them into a VoiceXML page for VoiceXML interpreter to extract.

The system engages with the user in multimodal interaction. It goes to the email channel to retrieve emails, perform email archiving and management in a dialogue fashion, read the email body, unpack and read email attachment for text and audio, respond the email to the sender by user’s request using recorded voice response as email attachment, etc. The main service logic in this application is kept at VoiceXML interpreter. User call the system and the system opens the email contact channel through hybrid-VoiceXML, gets email and generates a dynamic dialogue page based on newly retrieved email achieve. The dynamic page from the hybrid-construct will be issued to the VoiceXML interpreter. It engages with the user in dialogue according to the VoiceXML page from the hybrid-construct. This leads to new interactions and the subsequent dialogue will be generated to follow the inputs from the two interaction channels.

In order to compare approaches, we did a baseline implementation using multiple DCOM based extensions, which was a very complicated process. Comparing these two different approaches, the hybrid-VoiceXML based approach has several unique advantages. First, it is not tied to the platform as the other way does. Second, the hybrid-construct in hybrid-VoiceXML is a distributed server type resources. Once it is built, it is reusable and can be integrated with multiple highly different platforms. Third, the structure of hybrid-VoiceXML is a much modularized and clean integration for multiple channels. The communication between the hybrid-construct and the voice channel is through a natural VoiceXML page interface. This makes the application highly portable and positioned for distributed applications.

4.2 VoiceXML and Natural Language Dialogue Integration

Natural language dialogue and finite-state dialogue are two different types of dialogues. VoiceXML as a dialogue mark-up language is designed to support finite state dialogue. Natural language dialogue differs from finite-state dialogue in many ways. The speech recognition part in natural language dialogue is typically based on a statistical language model such as a word or word class based bi-grams or tri-grams. The understanding part in natural language dialogue is often based on sophisticated NLP process, such as inheritance chain, co-reference, latent semantic indexing (LSI) [6] etc. From the dialogue system point view, two types of dialogues belong to two different dialogue modes and usually it is difficult to integrate them together.

The application considered in this study is natural language call routing (NLCR). It is a task of directing the user’s call to the appropriate destination within a call center, or providing auto-response or key information according to the user’s request using natural language dialogue. It has various applications in customer relation management. The natural language understanding and dialogue generation of our NLCR system are based on a vector space based semantic model. It uses inheritance chain and latent semantic indexing to understand the user’s request and engage in dialogue to disambiguate the user’s query [3,7]. All these features are not covered in the current scope of VoiceXML.
We implemented the natural language understanding and dialogue generation of our NLCR system as if they were coming from a different modality channel using the framework of hybrid-VoiceXML. We used <object> tag extension to expose a second natural language speech recognition engine on the platform so that it becomes an alternative to the default finite-state based recognition engine for the VoiceXML interpreter. Dialogue interaction is through the voice channel. When user contacts the system, a dialogue based on VoiceXML is engaged first until it leads to a NLCR task. Once it enters NLCR part of the dialogue, the natural language speech recognition is applied through the <object> tag at the VoiceXML interpreter side, and recognition results are sent to the hybrid-construct as a request for NLCR based dialogue interaction. The dialogue control is therefore exchanged to the NLCR dialogue server that issues new dialogue request to NLCR DA. The DA generates a VoiceXML page according to the request and sends it to the server page of the hybrid-construct. The page is then provided to the VoiceXML interpreter. From user side and also strictly speaking, it is still VoiceXML engine that conducts dialogue. What is different though is that the dialogue comprehension, dialogue generation and the control are all from NLCR server at the hybrid-construct. At each dialogue turn, the dialogue action from NLCR is formatted into a page for VoiceXML interpreter to execute. The execution at the VoiceXML engine is therefore a slave process with the master process at the hybrid-construct. This process continues, until the NLCR process exits and then the dialogue control is back to the original VoiceXML based dialogue.

![Image](image_url)

Figure 3. Hybrid-VoiceXML implementation of NLCR

This study shows that the proposed hybrid-VoiceXML approach not only can apply to integrate other modality channels but also can conduct different type of dialogue in a uniform fashion even on the same voice channel. The natural language understanding and dialogue generation part in the hybrid-VoiceXML approach become reusable and distributed components. They can be shared by different platforms through HTTP interface. The natural language part at hybrid-construct can be made quite light, since most resources, such as TTS, ASR, tele-server, etc. are physical resource tier dialogue platform resources. However, the way to access these resources in our approach is quite unique. It is not through typical customized application API calls. It is through dynamic VoiceXML pages. The actual calling and management of these platform functions are done through the interpreter and by VoiceXML engine.

5. DISCUSSION AND SUMMARY

In this paper, a distributed multimodal dialogue system architecture was proposed based on the approach of hybrid-VoiceXML. It utilizes a special hybrid-construct to integrate multiple multimedia, multimodal processes into one dialogue that includes VoiceXML as its voice modality. Hybrid-construct in our approach has several important functions. It provides an additional abstraction layer for dynamic dialogue generation, which can greatly improve the efficiency and flexibility of the dialogue system. The dialogue agent layer developed for hybrid-construct has a server page layer at one end and DA at the other to synchronize and manage the interaction from each channel. Dialogue control between each interaction channel can be exchanged through the interface of a dynamic XML page. Several case studies were performed. It indicates that the proposed hybrid-VoiceXML approach is highly extensible. It can be used to form platform independent and distributed extensions for multimodal dialogue interaction beyond voice.

However, multimodal interaction is a fast growing research area presenting tremendous technical challenges. The study conducted in this paper is only a first step towards understand the realm of dialogue system design for multimodal interaction.

6. REFERENCES