EVITA-RAD : an Extensible Enterprise Voice PorTAl – Rapid Application Development tool

Yu Chen

Speech and Dialogue Processing Lab
Institute for Infocomm Research, Singapore 119613
yuchen@i2r.a-star.edu.sg

Abstract

Speech interactivity and call automation technology has reached a point of reliability sufficient for deployment at the enterprise level. Even though Voice User Interface (VUI) applications are growing in popularity, the development process remains a bottleneck to its widespread adoption. Currently available development tools are meant for either professionals or programmers experienced in VoiceXML, speech recognition, dialogue design, and platform specific languages. We present a web-based tool, EVITA-RAD, driven by an extensible and modular design paradigm for building dialogues. By incorporating database access and abstracting many issues of programming in VoiceXML into modules, our tool allows users to focus on designing dynamic and useful dialogue applications. Thus, EVITA-RAD lowers the barrier of entry for VUI application development, and helps promote widespread adoption of VUI applications.

1. Introduction

In spite of the benefits and the maturity of speech technology, VUI applications have not gained widespread adoption. Only enterprise call centers and reservation services have begun to use VUI solutions to automate operations and decrease costs. A well-known hurdle to the development of voice applications is that crafting a user-friendly dialogue is a painstaking process requiring multiple specialists. In addition, voice application deployment necessitates high end server machines and specialized hardware such as telephony cards.

A number of tools are currently available in an effort to lower these barriers of entry. Desktop IDEs for developing and testing VoiceXML applications, such as Nuance V-Builder[6] and IBM Websphere Voice Toolkit[3][3], provide many tools and libraries to aid in development and testing, but are aimed at experienced programmers with access to high end computers and deployment server equipments.

Simpler desktop IDEs such as CSLU’s RAD tool[8] and SUEDE[4] are available as free downloads. Though they serve as useful alternatives for novice users, they focus more on interface design and dialogue prototyping, and are not geared towards the development and deployment of industry standard VUI applications.

The most widely used environments are online developer networks and tools freely available from several voice application solutions providers. Members develop and test VoiceXML applications hosted in the web-based environment, then open a limited number of call-in extensions over these providers’ toll free telephone networks.

On the simpler end of the spectrum, a site such as Voxeo’s Developer Community[11] gives a downloadable tool for designing dialogue flow which then generates the corresponding VoiceXML code. Its few available online tools include an audio prompt recorder and a VoiceXML browser that runs the generated files and logs all calls. Developer sites like Tellme Studio[9] and VoiceGenie Developer Workshop[10] require the user to directly program in VoiceXML in their web-based environment, and thus provide the necessary programming tools such as a VoiceXML validator, and grammar building tools. Tellme even furnishes java applets with which the user can interact with the VoiceXML application via text input. Of the available online environments, BeVocal Café [1] has the most extensive set of tools, even allowing for multiple methods of text interaction to debug the application flow, tracing of program variables and server logs to aid debugging during an active voice call, audio playback of recorded calls, and a tool to estimate the number of phone lines needed to handle a given predicted traffic.

Web-based environments are easily accessible and provide hosting services and debuggers, but they do not provide interface design tools. These environments still require users to program and debug directly in VoiceXML, even though the ability to build applications without knowledge of VoiceXML has proven to be very useful[5], especially since novice users find programming complex tasks such as mixed-initiative forms and confirmations extremely difficult[2]. In addition, since VoiceXML is a markup language, it does not allow for developing complex, scalable or dynamic applications. Those features are only achievable through the addition of database access, so far unsupported by any development environment, and scripting languages such as JavaScript and JSP, only allowed for in the professional IDEs.

To address these deficiencies, we present a web-based VUI application development and deployment tool suite which adopts a modular and extensible framework for building applications. The complexities of programming directly in VoiceXML, using scripting languages and adding database access are abstracted into a set of flexible plug and play modules, seen by the user as an easy to use web interface. A series of managers are available to aid the user in debugging and tuning these and other components of the application.

2. System Description

2.1. System Design

EVITA-RAD is a web interface tool allowing developers to build VoiceXML based applications using predefined...
system modules. These modules are presented to the user as dynamic webpages which respond to input data. In addition, a series of managers are available for tuning different components of the application. Our system modules are implemented in JSP (Java Servlet Pages). Figure 1 shows how one example system module is composed of multiple JSP calls. These modules dynamically generate a block of VoiceXML code based on their input parameters, which executes the desired functions and returns output values. They are accessed by the main application as one or a group of subdialogs. The VoiceXML application is thus constructed by sequencing multiple subdialogs with a series of actions. As seen in Figure 2, the flow diagram of a sample application, the blocks (modules) are connected by arrows (actions), and these actions may be executed based on certain conditions. This programming structure emulates the object-oriented programming methodology of software development and is designed for flexibility and scalability.

![Diagram](attachment:image.png)

**Figure 1:** The Normal module is composed of several JSP functions. Most modules only contain one.

### 2.2. System Modules

We modeled system modules on either VoiceXML elements or functional calls to ensure that they are generic enough to build various types of applications[7]. Each module is seen by the user as one or more dynamic webpages consisting of forms, checkboxes, or drop-down menus. The module’s webpage views may not correspond exactly to its JSP composition. Forms on these pages are quite flexible, as the user can either input values or select from a list of application variables. EVITA-RAD contains six main modules.

- **The “Normal” module** is similar to the <form> element in VoiceXML. The user can fill in a prompt, grammar, confirmation parameters, and different exception handling actions, and can choose to enable skiplist and n-best recognition results. If a mixed-initiative grammar is used, a series of subpages will be generated to handle all the slot-filling probabilities. The output of the module is the recognition result.
- **The Database Query** module allows the user to fill out a simple or complex database query statement, which then returns the query result as an object.

- **The “Database Update” module** allows the user to update their personal database with static values, application variables, or results of previous modules or functions.
- **The “System Function” module** consists of a selection of functions which handle the Database’s date and time data formats or performs simple data manipulations. The results can then be used or saved within the application.
- **The “Confirmation” module** allows the confirmation of any application variable, be it a single value or an list of values. This variable does not have to be a recognition result. For example, a database query result can be used as long as it’s a single array object.
- **The “Call Transfer” module** is identical to the <transfer> element of VoiceXML. Transferring to a phone number effectively exists the current application.

### 2.3. Actions

Actions are available to all modules after the modules have successfully executed, or when exceptions occur. Actions allow the user to perform simple logic manipulations and branch the program flow to other modules.

- **Variable Declaration and Assignment** allows variables to be stored, edited, and accessed anywhere within the application.
- **Notification** is a flexible form for reading back text or application variables values, as well as audio prompts whose filename may be static or based on application variable values.
- **Goto** allows for branching to different modules given the fulfillment of predefined conditions.
- **Exit** plays the last exit message and ends the program.

### 2.4. System Managers

EVITA-RAD contains four main managers to help the user tune application properties separate from the application development process.

- **The “Database Manager”** allows the user to create, edit, and delete personal databases which can be accessed by any of their applications.
- **The “Grammar Manager”** allows users to create, edit, save, and share personal grammars and edit system grammar files, as well as generate dynamic grammar from their databases. The manager also guides users in creating top level mixed-initiative grammars and individual slot grammars.
- **The “Prompt Manager”** allows users to upload, review, and delete audio prompts from the prompt library.
- **The “Dictionary Manager”** allows users to review and add pronunciations to the system dictionary. The user can check their application at any time to verify whether all words from their application exist in the system dictionary. If out-of-vocabulary (OOV) words are found, the user can add a user-defined pronunciation for the new word to the system dictionary.
3. Building a VUI Application

![V-Leave flow diagram](image)

3.1. Application outline

The advantages EVITA-RAD has over other web-based VoiceXML application development tools can be best seen through an example. Our website provides two step-by-step guides on building sample applications for download. These two guides provide instructions on how to build an automatic directory assistant and a leave application system, respectively. We implemented applications which allow for information access and browsing because these functionalities are popular in the research community for prototyping a wide range of voice application parameters. In this paper, we focus on the leave system, and only outline the advantages of building it using EVITA-RAD versus using other tools.

The V-Leave (Voice-Leave) system is a database driven application where employees of a company can call into the system, authenticate themselves, check their available leave days, and submit and review leave applications. If managers call in, they will have the additional task of approving or rejecting the leave applications of their subordinates. This application incorporates many complex functionalities which cannot be handled by other development environments, such as database editing, date and time handling, mixed-initiative dialogue, and multi-slot confirmation. The application flow diagram is pictured in Figure 2.

3.2. EVITA-RAD advantages

The Normal dialogue modules (oval blocks) can be separated into three categories: static prompt(s), dynamic prompt(s), and response-confirmation. Dynamic prompts, such as the prompt reading back pending leave applications’ statuses, requires a loop mechanism to read back each submitted application, as well as methods to access individual elements in the database query result object based on the loop variable. In addition, the programmer may want to playback a set of audio prompts instead of using TTS (text-to-speech). All of this can be accomplished quite easily in the EVITA-RAD interface, using the Variable Declaration and conditional Goto actions to create a loop, while the Notification action can read back the variable values or array elements either through TTS or audio prompts with dynamically generated filenames.

All cases of response-confirmation can be handled by the flexible Normal module. Its web interface changes to reflect whether the module grammar is of the single-slot or mixed-initiative format, allowing the programmer to handle single or multiple slot recognition results and confirmations as needed. In either case, the programmer can enable n-best recognition results and skip list by selecting a checkbox. The confirmation will automatically adapt to the number of recognition results, as well as manage the skip list and re-prompts in case of any exception or zero accepted results. Even if the programmer never specifies any exception handling parameters, a default set of prompts are already enabled.

EVITA-RAD allows users to easily implement these complex elements, while other development environments do not provide help for mixed-initiative forms or exception handling. Mixed-initiative interaction is too complex for any inexperienced developer to create from scratch, but because it greatly enhances the flexibility and ease-of-use of a VUI application, we felt the necessity to develop a simple interface to allow novice developers to create robust mixed initiative interaction applications. Through experience building VUI applications from scratch, we also know that exception handling and confirmations are crucial but tedious tasks of building speech applications, especially when mixed initiative is used. Therefore, we’ve built in default exception handling and dynamic confirmation which automatically adapt to the module format, saving users an enormous amount of work over other web-based development environments.

The rectangular and diamond blocks in the flow diagram represent backend function modules such as database access and associated data handling. These features also do not exist in most other development environment. Only professional IDEs allow the user to build such functionalities from scratch, while web-based tools do not provide any database capabilities. In the V-Leave application, we are required to perform multiple database actions, such as querying the number of leave days available, creating a leave application and saving it to the database, and querying the status of submitted applications. These actions can be achieved through

---

1. [http://ivrad.i2r.a-star.edu.sg:8080/index.htm](http://ivrad.i2r.a-star.edu.sg:8080/index.htm)
our “Database Query” and “Database Update” modules, as well as the “System Function” features, which convert date and time data between the VoiceXML Browser format and the Database format, calculate the number of days between a start and end date, and calculates an absolute date given a natural language grammar string such as “today” or “next Wednesday”.

Given these features of EVITA-RAD, developers not experienced in VoiceXML are able to create flexible and scalable database driven VUI applications, simply through the sequencing of system modules which abstract away much of the details of programming in VoiceXML. Other development environments are aimed more at helping experienced VoiceXML developers debug and deploy static applications. In comparison, EVITA-RAD allows even novice developers to create and deploy complex and dynamic VUI applications.

3.3. User Testing

The current EVITA-RAD tool has been used internally by five experienced VoiceXML developers, and externally by three college students from Singapore Polytechnic who have no background in building dialogue applications. The five developers have had experience using Nuance V-Builder, while the students have compared our tool with IBM Websphere’s Voice Toolkit. After trying both IBM Websphere and the EVITA-RAD tool, the students decided to work exclusively on EVITA-RAD for their term project.

The overall response from the developers was positive, citing the simplicity gained with this tool as the main factor. A form which required up to an hour to complete now can be finished in a matter of minutes. The other main advantage cited was the fact that the dialogue flow is now much easier to understand and thus debug. Instead of wading through lines of VoiceXML code, the user can now use the web UI to focus on the application logic and dialogue module properties.

On the other hand, suggestions for improvement were made regarding the web interface, particularly that the input format and separate managers were initially confusing, and that program logic and flow was difficult to understand from the dialogue view.

Several other secondary institutions in Singapore who have seen demos of our tool have also expressed a desire to use it as part of their students’ coursework. Such positive feedback is very encouraging, and we expect future experiments to help us in improving the system.

3.4. EVITA-RAD Hardware Setup

The EVITA-RAD tool is deployed on the Nuance platform, on two Windows 2000 Professional computers with 3GHz Pentium processors and at least 512MB of RAM. The server machine runs Microsoft SQL Server 2000 and the Nuance Vocalizer 3.0 text to speech server.

The other computer hosts the actual RAD tool. Its configuration includes Java Development Kit (JDK) 1.3, Tomcat server 4.1, Nuance Voice Web Server 2.04, and the SIP Audio Provider Connectivity Pack 2.1.

Various software SIP phones are available for free download on the internet. The SIP address to dial into is vws@192.122.139.32:5060. By allowing for dial-in by SIP enabled VoIP hardware or software phones, we allow anyone around the world with an internet connection to dial in and test their applications, while other web-based environments’ 1-800 numbers are only geared towards users in North America.

4. Future Work

We plan to continue refining the UI and perform usability testing. We also intend to implement a framework of ‘user defined modules’ as an extension to the existing system modules. These user modules would allow a user to group together any consecutive set of system modules and their connecting actions, and define that as one user module. From then on, this user module can be reused as a virtual single module, with its own set of input and output parameters.

We also see many aspects of data input and manipulation that can be improved upon. For example, developers would appreciate linking to their own database instead of the one on our server, thus allowing their applications to retrieve real world data. Filters can also be provided to allow retrieval of data from outside sources, such as dynamic webpages.

5. Conclusion

Building VUI applications is usually an arduous task requiring high human and hardware resources. Current web development environments bring the ability to develop and deploy voice applications to the public, but are still difficult to use. A new generation of robust and user-friendly tools is needed to ease the high development hurdle and allow even novice users to create VUI applications. EVITA-RAD was created to meet these goals, and aims to bring the development and usage of VUI applications to a wider audience.

6. References