A Spoken Language Front-end for a Multilingual Music Data Base

Christian H. Schulz[†] **Dmitri Rubinstein**[†] **Dimitris Diamantakos**[†]
Michael Kaißer[†] **Jan Schehl**[†] **Massimo Romanelli**[†]
Thomas Kleinbauer[†] **Andreas Klüter**[‡] **Dietrich Klakow**[§]
Tilman Becker[†] **Jan Alexandersson**[†]

†DFKI GmbH
Stuhlsatzenhausweg 3
66123 Saarbrücken
Germany

‡Sonicson GmbH
Luxemburger Str. 3
67657 Kaiserslautern
Germany

§Lehrstuhl für
Sprachsignalverarbeitung
Saarland University
66041 Saarbrücken
Germany

Abstract: We present a VoiceXML-based spoken-language front-end for a multilingual music database. The multilingual setting poses new and unsolved challenges for the speech components and their integration. We discuss requirements on all components involved in such a scenario. After presenting the system architecture and the design of our dialog model, we discuss our experiences and finally point to future directions.

1 Introduction

The recent success of the iTunes Music Store and the start of a bundle of competing download shops have changed the face of the legal music download market. Still, the digital-music business is just in its infancy; the amount of sales through digital distribution is marginal compared to the classic channels based on physical media. But revenues are rising and the music industry is well advised in seeking for alternative business models to keep their stakes.

The high importance of the user's demand for mobility is impressively proven by the success of Apples combined business model, based on the iTunes Music Store and the portable iPod music player [App04]. This demand for mobility points out the need for innovative new user interfaces, since mobile devices raise different usability requirements than stationary desktop computers.

In this paper we report on on-going work where we combine the insight into the future market and our interest in exploring the capabilites of VoiceXML as technology base for providing access to music through spoken language. Whereas there is an increasing know-

ledge in the limitations with VoiceXML, e. g., [HWS+03], it is still the most widespread standard for encoding voice-based applications. Since VoiceXML was proposed it has become a standard for voice applications in different areas, such as call center support, tele-banking, automated phone-based interviewing and weather information systems, e. g., [ZSG+00].

The work presented here is based on an ongoing project at DFKI, i. e., the development of a JAVA-based VoiceXML interpreter (see section 4).

Our work presented here is a first step towards combining new technologies for advanced music search with the spoken language capabilities of VoiceXML. Our system—VoiceBeagle—is a VoiceXML-based front-end for the BEAGLE[1] music database for interaction with spoken language. BEAGLE allows for convenient access to music (see figure 2) to a database with different kinds of music by using typed natural language and a mouse. The system extends the standard database capabilities in two ways. Firstly, it allows for so-called *phonetic fuzzy match* making it possible to retrieve information despite non-perfect spelling from the database. The second feature allows for the usage of so-called *modifiers*, such as "slow", "faster" etc. by analysing the music signal from the MP3 files using a classifier [BKN02].

In what follows, we discuss the requirements a spoken-language front-end poses on the different components included in the VoiceBeagle system. Special attention has to be payed to multilinguality for speech recognition (ASR) and speech synthesis (TTS) of multilingual entries. Whereas our system is conceived for German users, the underlying data-base consists of music by, for instance, English artists. Even a database providing German artists only, poses the same challenges, since many titles and even names are taken from the English language.

The paper is organized as follows: In the next section we present the BEAGLE system. We discuss the requirements for a system allowing for spoken-language based access to the BEAGLE system in section 3 and provide a sketch for our prototype system and our architecture in section 4. Section 5 describes the dialog model and in section 6 we discuss our experiences so far. Finally, we conclude the paper and provide future directions in section 7.

2 Beagle

BEAGLE [BKN02, BK02] realizes the well informed virtual sales person as the central contact person for music seeking users in the inter-/intranet and mobile scenarios.

The combination of natural language queries and orthographic fuzzy matching offers simple and convenient usability. The automated audio- and similarity analysis functionality delivers optimal search results and the possibility of personalization takes care of personalized music recommendations.

BEAGLE can be integrated into any system where users access digital music archives,

[1] See `www.sonicson.com`.

such as internet CD shops, digital download offers, music burn stations or even corporate-internal music archives. Through its natural language processing capabilities in combination with a VoiceXML interface, the product can be used in all cases where an operation by mouse or keyboard is not possible. This is the case for mobile consumer devices such as portable media players, PDAs and mobile telephones as well as products of the electronic entertainment industry in stationary operation, e. g., hi-fi set-ups, televisions with internet connection or even in mixed forms of these: car radios with MP3 contents or internet access.

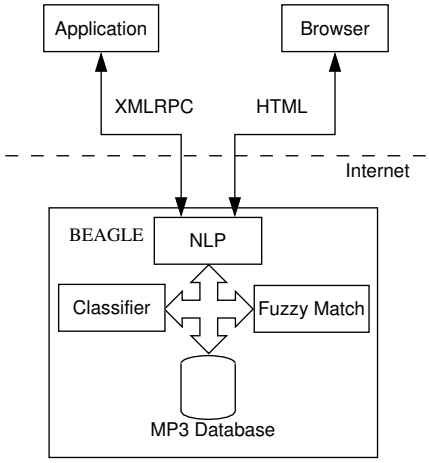

Abbildung 1: The BEAGLE architecture.

The system realizes a seamless combination of three major functionalities (see figure 1):

Natural Language Processing (NLP) Requests in natural language enable the music seekers to express themselves in the way they are used to (English words (in *italics*) are mixed into the German input):

- "Ein älteres Album von *Toto* mit dem Lied *hold the line*" (English: *"an older volume from Toto containing the title hold the line*)"

A high level of expressiveness can for example be obtained by using constraints:

- "hard rock, but only ballades with English titles"

Familiar "simple" requests for artists, titles or volumes can also be satisfied.

Integrated Phonetic Fuzzy Match The phonetic fuzzy match is an integrated component of the BEAGLE package extending the power of pure orthographic matching techniques. The system behaves robust in case of a wider range of typing errors, expands abbreviations, processes synonyms and is able to extract "sound-alike" database entries. Thus, queries like

- "El Jaro" (Al Jarreau),

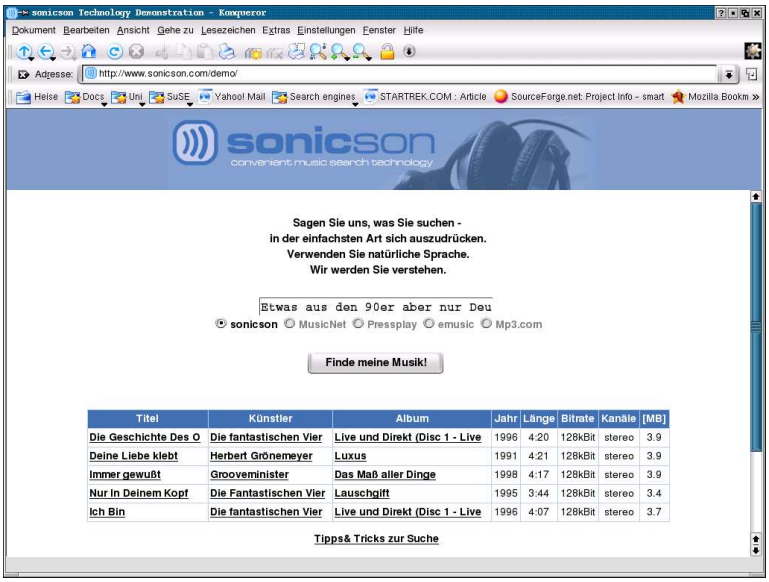

Abbildung 2: The German version of the BEAGLE HTML-based interface. The request *"Etwas aus den 90er aber nur Deutsch"* (Something from the 90s but just German) results in 5 German titles from 3 different artists. Note that "90er" (˜ "90th") is misspelled but the request is still analyzed correctly. Also note that the titles by "Die fantastischen Vier" stem from 2 different records.

- "Erey Kabadu" (Erikah Badu), and
- "You Two" (U2)

can be answered correctly. Note that this functionality is not used in the current VoiceBeagle system (see section 6).

Audio Analysis and Similarity Detection The BEAGLE core contains a powerful music analysis module. A fully automated classifier generates for example genre associations and examines music titles of the whole database for similarities. Thus, BEAGLE can fulfill requests for the following examples:

similar songs: "something similar to Madonna's last hit"

for mood: "a title to relax for the evening" and "lovesongs"

genre: "some soul for chillout" and "pop of the 90s"

Figure 2 shows the German version of BEAGLE's html-based interface. The user types a request in the text field and the system answers by displaying a page as shown in the figure. Possible follow-up steps include typing a new request and/or clicking on one of the links in the answer. By clicking on a title (Titel) the user activates a player. Important for this paper are the following observations:

- The user interacts with the system by typed input and mouse clicking.

- The system restricts the answer to 5 items even when the answer consists of additional items.
- The answer to a request includes links to for example record (Album), artist (Künstler) and title (Titel) which can be accessed by clicking on the respective link with the mouse.

Such design solutions are valid for interaction with keyboard and mouse. However, they are partly invalid for interaction with voice only. For example, the visual information including links for each item can impossibly be conveyed using synthesized speech. Therefore, alternative strategies have to be explored.

3 Requirements

As we saw in the previous section, the application system was designed for use with typed natural language in combination with mouse interaction using an internet browser. Some design solutions for example providing a lot of information for one item in the answer, are valid for interaction with the mouse. However, providing this information using spoken interaction would slow down the dialog thereby making the system boring and useless. Furthermore, the data base consists of items—for example, artists and titles—from multiple languages. The current database consists of German and English entries which should be accessible by spoken language. Moreover, our system has to be able to synthesize the responses from the data base. Thus, the TTS system will have to support multilingual output.

To sum it up: Our system has to fulfill the following requirements:

Multilingual input The user has to be allowed to use a mixture of languages, currently German and English. Although the application language is German, artists and titles will sometimes be in English. Therefore, the speech recognizer has to be robust if English words, spoken by a non-native, are embedded into a German sentence.

Multilingual output As for input, the output of the system will be a mixture of German and English. Therefore, the TTS must support synthesis of multiple languages.

Flexible dialog modeling Basic requirements for the model are to cope with arbitrary large numbers of answers, but also not too lengthy system generated prompts. In particular, an experienced user will not use the system in case the system behavior remains inelastic.

Finally, we have one remark. In particular, the first item above is a challenging research topic since it is rarely the case for non-native speakers of English that English artists, titles etc. are pronounced correctly. The same way as linguistic analysis had to be made robust against typos, a speech recognizer for this scenario will have to be very robust against incorrect pronunciation.

4 Architecture

At DFKI, we are in the process of implementing a VoiceXML interpreter for the VoiceXML 2.0 standard [W3C03]—the **D**FKI **V**oice**X**ml interpreter—DVX. The main reason for developing a VoiceXML interpreter was and still is[2] on the one hand the lack of a publically available complete implementation of the VoiceXML 2.0 standard that could be used on any platform. On the other hand, we are interested in exploring extensions to pure VoiceXML, such as, multimodality and discourse modeling. Another topic of interest is the usage of VoiceXML on PDAs.

The DVX interpreter is complete except for tags concerned with the telephone. It allows for including external modules via the `object` tag. We have included the Rhino ECMA interpreter [ECM99] into the DVX which, in addition to interpreting ECMA scripts, is also used to implement variables. We have also implemented the recognizer part of the JSAPI interface capable of managing grammars in the java speech grammar format—JSGF[3]. Our experiences so far are positive; the interpreter is fast and works flawless under UNIX, Linux as well as Windows. One advantage with our implementation is its small footprint: less than 200 kBytes. This makes it possibale to use the DVX on PDAs, a topic we will explore in the near future. Of course, we will have to sacrify the usage of ECMA scripts. However, we believe that smaller ECMA script interpreters will be available in the future (see also [BH04]).

Today's state-of-the-art speech technology fails in meeting the requirements listed in the previous section. Whereas the area of multilingual TTS has started to provide systems capable of synthesizing speech in multiple languages in a reasonable quality, speech recognition technology is far from mature when it comes to speaker independent recognition of large-vocabulary multilingual utterances. In particular, if the requirement is to use a commercial speech recognizer the selection of components is rather guided by pragmatic criteria: use one that meets the requirements as well as possible.

As the focus of this work was on integrating a complete system, we decided to use commercial components for ASR and TTS. As the VoiceXML-interpreter requires JSAPI as the interface, the following strategy has been used:

JSAPI↔SAPI We used the software package cloudgarden[4] that translates between the two standards. This enables us to use any SAPI-compliant ASR or TTS system.

Speech Recognition There are a number of SAPI compliant speech recognizers including a number of good small footprint recognizers for command and control available on the market. As we have a server available, we ruled out recognizers in the latter class and rather went for recognizers with a larger acoustic model. "Dragon Naturally Speaking 7" from ScanSoft not only has a good out-of-the box performance but also allows to train the acoustic model to the specific user. However, most important

[2] There are, to our knowledge, no complete platform independent interpreters available. The publicvoicexml project is aiming for this, but still lacks ASR functionality.

[3] See `http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/`

[4] See `www.cloudgarden.com`

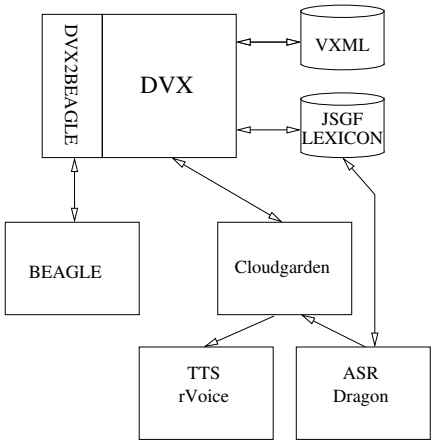

Abbildung 3: The VoiceBeagle architecture.

for this application was a simple access to the lexicon to tune it for recognizing non-German names and titles.

Speech Synthesis We use the *rVoice* synthesizer from Rhetorical which comes with a multilingual German and English option. As simple tags (XML-like markup) can be used to set the language even on a word level, it is not necessary to provide a phonetic transcription of the words of the ”guest-language”. As our data base presently sticks to these two languages, rVoice is a satisfying choice.

Our current version of VoiceBeagle is depicted in figure 3. The lack of telephone tags in our VoiceXML interpreter constrains our experiments to a desktop-based system.

BEAGLE offers an XMLRPC[5] interface which allows for convenient interfacing via the internet. For the interpreter, we have programmed an extra JAVA class which is integrated into the dialog model via the `object` tag. In addition to the pure communication functionality, the class provides functionalities for inspection of data-base requests as well as converting the data-base answers to input for the rVoice synthesizer. A player for listening to the music is controlled via ECMA scripts [ECM99].

One additional advantage with our JAVA-based interpreter, is the ease with which the integration of external packages such as XMLRPC can be done.

5 The Dialog Model

Crucial for the design presented below is the decision to use the NLP capabilites of BEAGLE for analyzing the queries. Therefore, most of the user utterances are passed to

[5]See `http://www.xmlrpc.com/`

BEAGLE without linguistic analysis. Consequently, we do not know what the user asked in some situations. However, we can infer the query type by carefully examining BEAGLE's reply. Thus, our main work has been to process the results of the database lookups in an intelligent way. We hence classified the possible classes of answers by BEAGLE (see below).

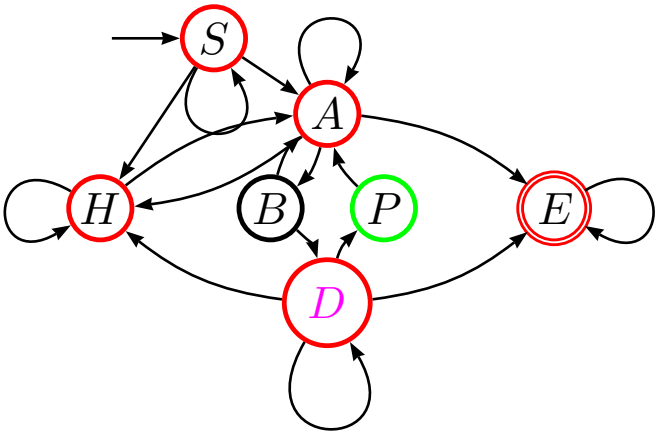

Abbildung 4: The dialog model. S = Start, A = User, B = Beagle lookup, D = database response handling, P = play, H = help, E = exit.

Our dialog model has been designed to meet the following requirements:

Functionality For the initial system, we allow for browsing the database and playing the results. Further possibilities discussed include constructing play-lists. This functionality, however, has been postponed into the future. More precisely, the application allows for

- querying the database
- play tunes, stop playing tunes, interrupt and exit

Abstraction On the other hand, we want to present the intended functionalities (especially browsing) as naturally as possible. To achieve this, some abstractions from the actual API of the application are necessary, e. g., the system responses are filtered and their presentation as well as the next step in the dialog depend on whether all five songs are by the same artist or not. See figure 5 for more details. Most of these changes are due to the fundamental differences between an HTML-based and a speech-based interface.

Dialog Style The ultimate goal of natural language based interaction is to be as natural as human–human interaction. Current VoiceXML-based interaction is quite limited and this must be signalled to the user. Thus, we allow on the one hand for a flexible mixed initiative dialog where the user can intervene in most situations and pose new

commands and/or requests although the system has requested information by the user. On the other hand, we use a system-driven approach in the design of the dialog with explicit system-initiated queries—as opposed to open-style contributions—to keep it in control of the dialog as much as possible.

Important was the eagerness to minimize the number of turns. The dialog model allows for shortcuts in some cases (see the sample interaction below). We deploy a quite exhaustive help system which tries to anticipate difficulties caused by the user and/or the system, such as

- offer help in dead ends
- guide the users for the purpose of their satisfaction

Language Style For the initial system we use quite verbose system contributions. We also allow the user to be both verbose and short using command-like utterances. The latter is because in many situations some words and/or expressions are unambiguous. Another reason for this is also that the users get used to the system after a short time.

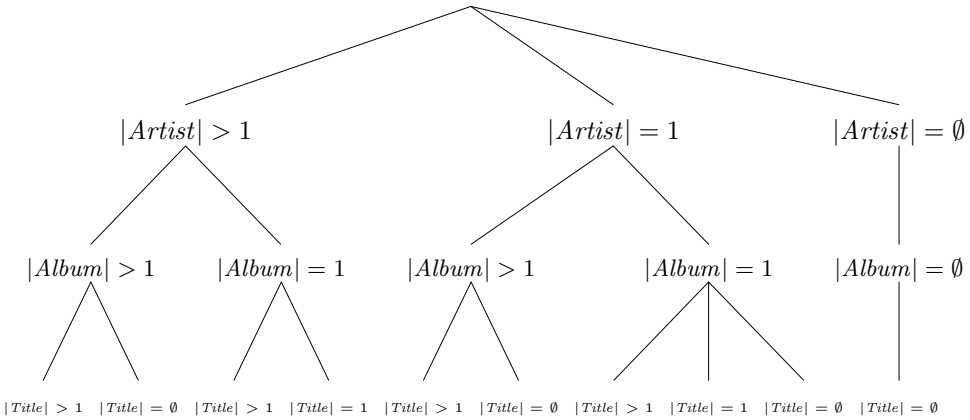

Abbildung 5: The decision tree for the analysis of the BEAGLE responses. $|Artist|$ is the number of distinct artists, $|Album|$ is the number of distinct record titles and $|Title|$ is the number of titles.

We provide a state-based visualization of our dialog model (see figure 4). However, the capabilities of our model goes beyond the strict state-based paradigm in that we allow the user to perform several actions in every situation. In general, it is possible in any state to access the help system. The start state is marked with "S". There, the system introduces itself and asks the user what he would like to do, i. e., what music request the user has. In case the user poses a music request, it is forwarded to BEAGLE (corresponding to state "B"). In state "B" the response is analyzed and—depending on the response—different transitions to the main part of the model (the D(ialog) state) are performed. The "D" state is implemented by 6 different dialog documents each handling different situations.

The analysis of the response is based on a decision tree depicted in figure 5, where we count 10 different paths corresponding to the relevant results returned by BEAGLE. For instance, the left-most path stands for the result structure with the most complex kind of information; in this case there are different artists, records, as well as titles. The right-most path stands for the result when none of the items in a query is valuable for BEAGLE. Hence, for all the possible results, there is a match with one of the cases in the decision tree, which further trigger the assignment to the appropriate dialog among the different dialogs in "D". Our current heuristics is to provide a maximum of 5 items in an answer.

In state "D", the result is presented and the user can either play, refine the question or pose a new one. Note that the user can exit or ask for help at any state.

The application consists of 25 different grammars with 138 rules in total. To reduce errors in speech recognition, we have trained the ASR with just a part of the complete database. For our initial tests, we used about 25 titles, 80 record names and 50 artists which required around 8 hours of training time. For a further reduction in ASR errors, we dramatically reduced the lexicon from the 250,000 entries offered by Dragon to only 200.

5.1 A Sample Interaction

Below, we provide a verbose version of a sample interaction with the system. The German original is included in the appendix. The underlined words/expressions indicate the command-like style accepted by the system.

VB13: ... before you start, do you want information about the details of the VoiceBeagle system or do you want to start with a music query directly?

U14: I would rather make a music query.

VB15: What music do you want?

U16: Are there any records by Miles Davis?

VB17: Either I could not determine which artist you want or the selected artist is not available. Do you want to make a new query or hear a selection of other artists.

U18: Preferably, I would like to have a small selection of artists.

VB19: Are you interested in an artist from the following selection: Phil Collins, Incognito, Brooke Russel, Prince or The Brand New Heavies?

U20: Yes.

VB21: Please state the name of the selected artist!

U22: Phil Collins.

VB23: ...

5.2 Discussion

As indicated by the underlined parts, most of the user and system contributions exist in shorter versions. It is, for instance, possible to say *"anfangen"* ("start") or even pose a request directly instead of the verbose version of **U14**. Additionally, instead of answering **VB19** with yes, it is possible to select the artist directly.

The request in **U16** cannot be met by BEAGLE since "Miles Davis" is not in the database. Instead, BEAGLE returns a list of alternative artists. This is recognized by our dialog model, i. e., decision tree, in that the response consists of just artists and no titles corresponding to the second left-most branch in the decision tree. In such a case, the user is given the opportunity to explore alternative artists.

We have also experimented with replacing **VB19** with a mixture of **VB19** and **VB21**. A possible alternative could be *"Say one of: Phil Collins . . . , or The Brand New Heavies, or make a new query."* (*"Sagen Sie einen von Phil Collins . . . , oder The Brand New Heavies, oder geben Sie eine neue Anfrage!"*) However, the design will be finalized after the system has been evaluated.

6 Experiences

The overall experiences with the system are positive and it is fairly easy to get a hold of and play some music. An initial evaluation showed that, compared to the HTML-based interface, the access via VoiceBeagle is classified as better. Due to the small number of test persons, this statement is only indicative and has to be verified in a real evaluation in the future. There are some details and obstacles though which limit the "fun factor" and thus the success of the system. For some of them there are work-arounds but some of them have developed to real obstacles or challenges.

Below, we provide a list with experiences, starting with the positive ones and drift over to the challenges lower down in the list.

Dialog Model As indicated in section 5, the model has advantages but the final structure has not been set. This is mainly because of the lack of a bigger set of users. Our initial tests show a good acceptance although some of the restrictions in the dialog model were regarded as inconvenient.

Speech Recognition While the Dragon speech recognizer proved to be a good choice in that it allowed for the training and recognition of non-standard speech, it is not useful for speaker-independent recognition. The limitations in vocabulary is a real hindrance; we really do not know how the system works when the user has access to a bigger database. A natural and necessary next step will include the evaluation of alternative speaker-independent speech recognizers.

Speech Synthesis This is probably the most satisfying external component. While still not always natural, it synthesizes all sentences in such a way that understandability

is very high. As always, one has to get used to the voice and in particular the way the voice pronounces mixed language. However, the TTS voice always speaks in the same tempo which can be, after using the system for a while, a little annoying. This has to do with the fact that the users quickly get acquainted with the system and thus know what is coming. Possible solutions are: faster synthesis for experienced users and shorter prompts or a combination of both.

Language Analysis should be integrated in the interpreter. This makes it possible to understand if the requested item is in the database or not. Additionally, we can produce more precise prompts. These would enhance the acceptance of the VoiceBeagle.

Note that the phonetic fuzzy match is not used at all since the ASR outputs correctly spelled artists, titles etc. from the ASR lexicon. We believe, however, that in an extended version of the system, it will not be possible to include all names from the music database in the ASR lexicon. In this case, access to phoneme level analysis of the ASR component could be combined with the fuzzy match component to allow for arbitrarily large databases.

Discourse Model VoiceXML itself does not provide mechanisms for maintaining the state of the current discourse. To a limited extent, we have implemented the discourse context through the use of variables and the object tag. It would be desirable to have standard mechansims for effects like pronomina resolution, elliptical utterances, current discourse context etc.

Software Infrastructure In particular the combination of Cloudgarden and Windows 2000 has evoked annoyingly many system breakdowns. These do not occur that frequently with Windows XP. In future versions of the system, we will use alternate approaches to connect the VoiceXML interpreter to the speech components.

7 Conclusion and Future Work

We have presented a VoiceXML-based dialog system for accessing a multilingual music database using spoken language. Initial tests have shown that our approach to realize a voice-based access is plausible and convenient. While our initial dialog model is expandable in several directions, the core of the system has received a positive acceptance. One of the major obstacles is speech recognition and to a lesser extent also speech synthesis: multilinguality and foreign language pronunciation is a challenging topic.

We believe that the global development of music download, e. g., Apple's iPod and iTunes, shows that there is a great interest in such services. We will continue our efforts in multiple directions which will include the following topics:

Interpreter Parallel to our application, we will continue to develop the DVX. The first track is to extend it to cover the complete VoiceXML 2.0 standard. In particular, we are aiming for the telephone tags. Since our interpreter has a very small footprint it is possible to run it directly on a mobile device. However, since a large-vocabulary

speech recognizer as well as synthesis will have to run on a server, such applications will, at least initially, be small.

Speech Recognition We will evaluate the market for large-vocabulary speaker independent speech recognizers. The main challenge is to cope with non-natural pronunciation of items in the foreign language, i. e., English. Our current version does not make use of the fuzzy match functionality of BEAGLE core. This does not make sense since the output from our ASR component is always correctly spelled. However, when scaling up the system, it is not possible to train on the entire content of the data base. Moreover, the pronunciation of foreign language increases the degree of errors in speech recognition. Indeed, for smaller applications like VoiceBeagle, today's speech recognition technology allows for the identification of regions in the speech signal which contain—in our case—an artist or title. Possible future work include investigating how phoneme-based output from the speech recognizer can be used to match the content of the database in the same spirit as for the current version of BEAGLE.

Telephone The previous item makes it possible to realize a telephone-based version of VoiceBeagle. Today's and tomorrow's mobile devices come with enough processing power to run parts of the application directly on the mobile device. We will investigate how this can be achieved.

Linguistic Analysis As noted in section 6, we will integrate the linguistic analysis of Beagle directly into DVX. This allows for more precise and intelligent handling of misunderstandings.

Discourse Memory VoiceXML lacks a generic model for treating of discourse information. Consequently, the implementation of request–response pairs involves much handicraft. We have started to investigate if and how generic techniques like those suggested in the SmartKom project [Wah03, PAB03, PEA03] can be incorporated into VoiceXML.

Dialog Strategies and User Adaption As indicated by our initial evaluation, we will need to adapt the dialog strategies for experienced users. There are several points we would like to address:

- Shorter answers for experienced users.
- Faster synthesis for experienced users.
- It is, for the experienced user, plausible to define personal shortcuts. These could, for instance, play the favorite tunes or private compilations.

We will additionally have to deploy flexible means of understanding when and how this occurs and then develop techniques for dealing with these effects.

Our work leads us to the conclusion that VoiceXML is mature enough to implement end-to-end spoken dialog systems efficiently. For the further development of such systems, we have identified a number of challenges for speech technology, e. g., multilinguality and

for dialog modeling, in particular the inclusion of discourse processing. Finally, we have shown that the combination of innovative existing technology with speech via VoiceXML provides an exciting entry point to the rising market of digital music access.

Literatur

[App04] Apple. *iPod*. `http://www.apple.com/ipod`, 2004.

[BH04] Dirk Bühler und Stefan E. Hamerich. Towards Embedding VoiceXML Applications Through Compilation. In *This Volume*, 2004.

[BK02] Stephan Baumann und Andreas Klüter. Super Convenience for Non-Musicians: Querying MP3 and the Semantic Web. In *Proceedings of the ISMIR-2002*, Paris, France, October 2002.

[BKN02] Stephan Baumann, Andreas Klüter und Marie Noriel. Using natural language and Audio Analysis for a Human-oriented MIR System. In *Proceedings of the Second International Conference on WEB Delivering of Music (Wedelmusic 2002)*, Darmstadt, Germany, December 2002.

[ECM99] ECMA International. ECMA-262 ECMAScript Language Specification 3^{rd} Edition, December 1999. `www.ecma-international.ch`.

[HWS+03] Stefan W. Hamerich, Yo-Fang H. Wang, Volker Schubert, Volker Schless und Stefan Igel. XML-Based Dialogue Descriptions in the GEMINI Project. In Robert Tolksdorf und Rainer Eckstein, Hrsg., *Berliner XML Tage 2003. (CFP und Programm) Proceedings zur Konferenz*, Seiten 404–412, Humboldt-Universität zu Berlin, Oktober 2003.

[PAB03] Norbert Pfleger, Jan Alexandersson und Tilman Becker. A robust and generic discourse model for multimodal dialogue. In *Workshop Notes of the IJCAI-03 Workshop on "Knowledge and Reasoning in Practical Dialogue Systems"*, Acapulco, Mexico, August 2003.

[PEA03] Norbert Pfleger, Ralf Engel und Jan Alexandersson. Robust Multimodal Discourse Processing. In Kruijff-Korbayova und Kosny, Hrsg., *Proceedings of Diabruck: 7th Workshop on the Semantics and Pragmatics of Dialogue*, Wallerfangen, Germany, September 2003.

[W3C03] W3C Candidate Recommendation. *Voice Extensible Markup Language (VoiceXML) Version 2.0*. `http://www.w3.org/TR/2003/ CR-voicexml20-20030128/`, 28 January 2003.

[Wah03] Wolfgang Wahlster. Towards Symmetric Multimodality: Fusion and Fission of Speech, Gesture, and Facial Expression. In A. Günter, R. Kruse und B. Neumann, Hrsg., *KI 2003: Advances in Artificial Intelligence*, number LNAI 2821 in Lecture Notes in Computer Science, Seiten 1–18, Hamburg, Germany, September 2003. Berlin, Heidelberg: Springer.

[ZSG+00] V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T. Hazen und L. Hetherington. JUPITER: A telephone-based conversational interface for weather information. *IEEE Trans. on Speech and Audio Processing*, 8(1):85–96, January 2000.

Appendix

The German original of the interaction as presented in section 5.1.

VB13: ... Möchten Sie, bevor Sie anfangen, das VoiceBeagle System kennen lernen, oder direkt nach Musik suchen?

U14: Ich möchte direkt nach Musik suchen!

VB15: Welchen Musikwunsch haben Sie?

U16: Gibt es Alben von Miles Davis?

VB17: Entweder konnte nicht festgestellt werden, nach welchem Künstler Sie suchen oder der von Ihnen gewünschte Künstler ist nicht verfügbar. Möchten Sie eine neue Anfrage stellen oder hätten Sie gerne eine Auswahl von verfügbaren Künstlern?

U18: Ich hätte gerne eine Auswahl von Künstlern.

VB19: Interessiert Sie aus folgender Auswahl ein bestimmter Künstler: Phil Collins, Incognito, Brooke Russel, Prince oder The Brand New Heavies?

U20: Ja.

VB21: Bitte nennen Sie den gewünschten Künstler.

U22: Phil Collins.

VB23: ...