Architecture for adaptive multimodal dialog systems based on VoiceXML

Georg Niklfeld(1), Robert Finan(2), Michael Pucher(1)

(1) Telecommunications Research Center Vienna (ftw.)
(2) Mobilkom Austria AG
`niklfeld@ftw.at, r.finan@mobilkom.at, pucher@ftw.at`

Abstract

This paper describes application oriented research on architectural building blocks and constraints for adaptive multimodal dialog systems that use VoiceXML as a component technology. The VoiceXML standard is well supported and promises to make the development of speech-enabled applications so easy that everyone with general web programming skills can accomplish it. The paper proposes a software architecture for multimodal interfaces that emphasizes modularity, in order to strengthen this potential by clearly separating different types of development tasks in a multimodal dialog system. The paper argues that adaptivity is a central design concern for multimodal dialog systems, but that adaptivity is not facilitated by the current VoiceXML standard. This and other examined limitations of VoiceXML for building multimodal dialog systems should be repaired in upcoming work on a successor standard that will explicitly target multimodal applications.

1. Introduction

Multimodal interfaces are important for pending 3G telecommunication data services, which will be invoked from devices that have speech input/output capabilities but only small touch-screens. The VoiceXML standard [1] is important in this context because it promises a streamlined development process for speech-enabled applications, which will allow third party software developers without much speech processing expertise to create the wide range of applications that is needed to make 3G and speech-access attractive to consumers. A broad array of adaptation techniques are required, because in mobile usage environmental conditions vary greatly, and because multimodal interfaces have many open parameters that need to be set for optimum usability in specific circumstances. In this paper we describe design work on a multimodal dialog system prototype developed by our group which attempts to reflect these issues. The paper describes a set of building-blocks that emerge, but also shortcomings of the VoiceXML standard that should be addressed in future derivate standards.

2. Multimodality, adaptivity, standardized components

This section presents the arguments for the design goals of multimodality, adaptivity and the use of standardized components, which motivate the model architecture described in the subsequent section.

2.1. The importance of multimodality

Human face-to-face communication is multimodal, combining the acoustic channel with visual and occasionally tactile channels. Human beings are therefore well equipped to communicate multimodally, and multimodal communication is perceived as very natural. In human-computer interaction, the use of spoken or written natural language poses complexities that have led to a dominance of visual interfaces consisting of text and usually 2D graphics. Visual interfaces are very efficient under some circumstances: written text allows rapid information uptake; visual interfaces on large screens allow for showing large amounts of information simultaneously, and humans are able to focus on bits that interest them without having to process all the rest in-depth; finally, visual interfaces are preferred consistently for input and output of spatial information [2]. In visual interfaces, text output can be used for information for which graphical metaphors are not available, text input for unrestricted content or inherently linguistic information such as names.

Yet, considering data services on 3G mobile devices, the following factors constitute obstacles for relying solely on visual interfaces, and imply a real usefulness of added speech interface capabilities:

- The terminal devices where 3G data services run will have small displays in terms of size and resolution, although a significant improvement over current WAP phones is expected. Still, it will not be possible to mirror the user experience of todays commercial web-sites concerning the amount of information that can be presented simultaneously. (Note also that the speed disadvantage of TTS is less problematic for small amounts of information.)
- While 3G terminals will be produced in various form-factors, many of them will be too small to provide comfortable alphanumeric keyboards, relying instead on pointing devices such as a pen/touch-screen combination. Without a keyboard, text input becomes cumbersome, even when some handwriting recognition/graffiti technology is provided. Speech input is a logical alternative. The input dilemma of 3G devices is made yet more severe where even a pointing device is lacking, like in current mobile phones. This WAP-like scenario makes data services without speech support so unattractive that we consider it unlikely that large numbers of 3G devices intended for data access will be in this category.
- In mobile usage situations, visual access to the display may be impossible in certain situations, such as when driving a car. Also, even where pointing-device and keyboard are provided, access to them is not possible when a user has her hands busy for other activities. A speech interface may still be usable in such situations.

We contend that the combination of these considerations takes the case for multimodal interfaces for 3G services beyond a nice-to-have status to that of a significant building-block that needs to be put in place for successful deployment scenarios of 3G infrastructures to emerge. This is also the motivation for application oriented research like the one described here, which examines the technical feasibility of an economically attractive development model for multimodal interfaces.

2.2. Types of adaptivity

We distinguish four practically relevant types of adaptation tasks in multimodal dialog systems, as indicated in table 1.

	situation attributes	*user attributes*
user control	user-controlled situation adaptation	user-controlled user adaptation
system control	system-controlled situation adaptation	system-controlled user adaptation

Table 1: *Adaptation types*

The discussion in section 2.1 referred to the variability of usage situations for data services that comes along with mobile usage. We refer to this type of adaptive requirements as *user-controlled situation adaptation*. It demands user interfaces that allow a flexible choice of modalities at many points during interaction with the system. We believe that typically, the visual interface will be the primary interface, which is active at all times but allows the user to branch off into voice usage for a suitable chunk of the interaction.

System-controlled situation adaptation comprises techniques where the system autonomously monitors the interaction, attempts to reason about situation attributes, and adapts in order to avoid problems. For example, high levels of ambiance noise may lead to poor performance of speech recognition. The system may find out about this either by monitoring for low confidence scores in speech recognition, or by monitoring the frequency of corrections by the user. It may then choose either to deactivate speech input completely, or, in a system that can process coordinated, simultaneous inputs from the visual and speech modalities, it may assign relatively lower weights to the results from speech recognition and more to recognition results from the visual interface, e.g. gesture or handwriting recognition [3].

Under the item of *user-controlled user adaptation*, multimodal dialog systems should allow an explicit modality choice to suit a user's personal preferences. E.g. it should be possible to disable all speech-interface elements of the interface. Where user profiles are stored between sessions, the system should allow to configure an assignment of elements of the interface to one or the other modality, either through an explicit profile-configuration mode (e.g. a user might want to assign list-boxes to the speech modality), or by recording the modality choices of the user in the background during interactions. Again, where systems allow coordinated, simultaneous input on multiple modalities, the user should be able to select a relative weighting for the modalities.

Finally, *system-controlled user adaptation* is an important issue in speaker independent automatic speech recognition (ASR) systems as common in telephony-oriented dialog systems. As the available amount of adaptation data from a speaker grows during a session, iterative speaker adaptation can reduce the word error rate of a speaker independent system to about half, until the error rate of comparable recognizers that are trained in a speaker dependent fashion is matched [4].

Outside the acoustic domain, other relatively static characteristics of users can be reflected in user models [5] that can lead to adaptation of dialogs or user interfaces. Such user models collect data on: knowledge of the user, to influence the amount of guidance provided; abilities of the user, to influence the choice of more or less complex input patterns by the system; misunderstandings exhibited by the user, to provide tailored explanations of examples. User models can either be acquired new for each user during the interaction (although for telephony based dialog systems, which are typically used infrequently by any user, this may not be feasible), or an existing user model may be attributed to new users, and then adapted further.

While all four discussed adaptation types are important for good multimodal dialog systems, the subsequent discussion of available standardized components and system architectures, and in particular of the VoiceXML standard, will show that these frameworks currently do not support system-controlled adaptation adequately, especially not in those cases where information from speech recognition and from the dialog management platform would have to be integrated. We will return to this problem in section 5.

2.3. Standardized components

In 1999, the EU-sponsored DISC project undertook a comparison of seven toolkits for dialog management in spoken dialog systems [6]. The dialog management component is the central part of a dialog system and of particular importance in a discussion about streamlined development processes for multimodal interfaces, because it is the bridge between the intelligent interface technologies and the underlying application logic of application servers and databases. The DISC-survey finds platforms that offer rapid prototyping support, partly via integrated development environments. As most of the toolkits are shipped together with speech recognition and speech synthesis engines, this frees application developers to focus, firstly, on the dialog design, and secondly, on interfacing to the application core.

The work done with W3C support on the VoiceXML standard [1] represents another conceptual step forward from that state of affairs for two reasons. Firstly, being an internet standard, VoiceXML goes beyond manufacturer-specific toolkits that are not compatible with each other by providing a credible set of target interfaces for all players in the industry. Second, it chooses XML and the associated web-programming technology both as a format for specification of the dialog management, and for interfacing to the application logic. The two most important aspects for a standardized component framework for dialog systems have thus been brought in line with web technology, which is what was needed to create that promise of a platform for speech-enabled applications that is easily accessible to developers with a general, and not speech processing specific, background. Today one can add that VoiceXML has received important support from major players in the industry who have made development platforms for VoiceXML-based applications available free of charge for developers. The only potential downside of recent developments is that the platform manufacturers have also included some proprietary elements in their implementations, making direct transfer of VoiceXML applications from one platform to another again impossible.

While early requirements documents for VoiceXML explicitly treat multimodality in the context of the discussion on the inclusion of DTMF input in the standard [7], the standard is not written to cover general multimodal applications. Subsequently, W3C has drafted a requirements document for a multimodal dialog language [8] and currently a new working group on that topic is being assembled. At present however, the standard provides no intentional support for general multimodal systems. The most important drawback that we found in our attempts to nevertheless build a multimodal architecture *around* VoiceXML, is that it is not possible to make an active voice dialog running in a VoiceXML browser aware of events that occur outside the voice browser, e.g. at a visual interface: VoiceXML neither allows for linking in Java applets that could receive pushed notifications, nor does it provide any other interface for external events. Our architecture for multimodal dialog systems based on VoiceXML is considerable influenced by this fact.

3. Architecture

In a project at our institution that follows the longer term goal to develop architectures for multimodal dialog systems for 3G telecommunications infrastructures, for the benefit of our supporting partner companies from the Austrian telecommunications industry, we are currently developing an architecture that shall: use mainstream technologies and standards as far as possible, to test their capabilities and limitations; be general enough to scale from our first small prototypes to bigger systems that are close to market-readiness; provide the basis for usability research on multimodal interfaces.

The architecture shall support a multimodal interface that shall combine a visual interface via HTML and Java applets in a visual web browser with a voice interface built using VoiceXML. Communication between the visual browser and the voice browser is mediated via a central application server that is built using Java servlet technology in combination with a web server.

In [8], three types of multimodal input are distinguished:

1. *sequential multimodal input* is the simplest type, where at each step of the interaction, either one or the other input modality is active, but never more than one simultaneously;

2. *uncoordinated, simultaneous multimodal input* allows concurrent activation of more than one modality. However, should the user provide input on more than one modality, these informations are not integrated but will be processed in isolation, in random order;

3. *coordinated, simultaneous multimodal input* exploits multimodality to the full, providing for the integration of complementary input signals from different modalities into joint events, based on timestamping.

Because we cannot send events about the visual interface to the dialogs in the voice browser (cf. section 2.3), we maintain that only the *sequential multimodal input* pattern can be properly realized with the current version of the VoiceXML standard: The other patterns require that even while the voice interface is active, e.g. listening for user speech, it must be possible for the multimodal dialog to change state based on inputs received from the visual interface. In the sequential mode on the other hand, it is possible to deactivate the visual interface whenever voice input is activated.

In this case then, the choice of multimodal interaction pattern is determined by features of the components used. In many realistic application development efforts, the interaction pattern will be determined by user-level requirements that have to be met. Anyhow, the choice of multimodal interaction pattern will certainly be a dimension in which variation occurs. For the purposes of the demonstrator development in our project, it was important to find a software architecture that can remain stable across different patterns and across interface types.

This can be accomplished by a modular or object-oriented design which separates the central application server into the following functions:

visual communicator: a modular handler for the visual interface;

voice communicator: a modular handler for the voice interface;

transaction module: encapsulates the transaction needed for the application logic;

multimodal integrator: handles all message flows between the interface modules, and between the interface modules and the transaction module.

The resulting system architecture is shown in Fig. 1.

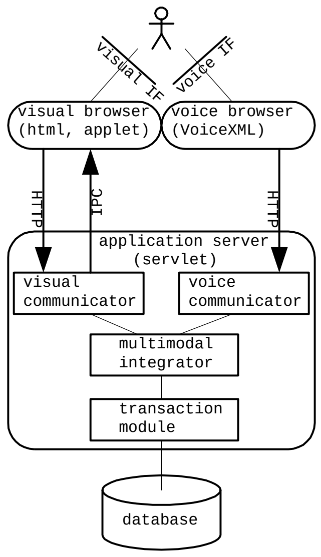

Figure 1: *Multimodal Architecture*

Both the visual and the voice user interface are realized by standardized components in line with general web technologies (visual web browser and VoiceXML browser). Within the application server, the architecture stipulates a specialized interface handler for each modality.

For example in our prototype, the *voice communicator* prepares VoiceXML documents which it puts on a web server that is associated to the application server. Once the VoiceXML interpreter of the voice browser has been started (after a user-triggered event, e.g. an incoming phone call), each of the VoiceXML documents processed by the interpreter is programmed so as to terminate with a load of a successor document from the web server. Our *voice communicator* simply prepares these successor documents based on messages it receives from the multimodal integrator. When the voice interface is not active, the prepared documents contain just an idle loop that terminates after some time, e.g. 0.5 seconds. When the *multimodal integrator* decides (based on user input) that a chunk of the interaction shall be performed via voice, it sends a message indicating field labels and data-types (e.g. the range of an enumeration) to the *voice communicator*, which instead of an idle

VoiceXML document now produces a VoiceXML dialog document that executes the respective voice dialog with the user and returns results to the *multimodal integrator*.

The *visual communicator* is designed similarly to prepare HTML pages and applets for the visual browser. In our prototype, the visual interface includes controls that allow the user to explicitly activate the voice interface for a group of input fields. When this is done, the *visual communicator* deactivates all control elements on the visual interface, and sends a message with the user request for voice dialog to the *multimodal integrator*.

The *multimodal integrator* is the only part in the proposed architecture where information from more than one modality is processed. The way this processing is done defines the multimodal interaction pattern. To change the pattern, the architecture envisages that one implementation of the *multimodal integrator* would simply replaced by another, without any changes to other parts of the systems. This of course presupposes that the interfaces between the the *multimodal integrator* and the interface handlers have been defined so generally that all occurring multimodal interaction patterns are covered. A description of this interface is an area for further study.

4. Application

We believe that within the 3G data service scenario discussed earlier, even with only sequential multimodal input it is possible to create multimodal interfaces that are better than existing interfaces that are just visual. Currently we are developing multimodal interfaces to existing web services with visual interfaces and plan to perform comparative usability studies. For our prototype, we target a typical route-finder application, where the user enters two addresses in Vienna and the system returns a graphical map that shows the shortest path between the two locations (such a service is available on the WAP portal of one of our partner companies). On mobile devices that do not provide a comfortable keyboard, entering street names is difficult. The range of possible names is also too large (c. 8000 for Vienna) to make selection from a list-box an option. In our interface, we request to specify the city district or the initial letter of the street name first. This can be entered by voice, or via selection from a visual list, and narrows down the search space for possible street names to 400 names on average. We assume that most users will prefer to use speech recognition for the selection among these 400. If speech recognition proves too unreliable even for that, then it is also possible to specify both district and first letter, which will narrow the search space down to c. 16 names.

5. Discussion

In our view, these preliminary results show that VoiceXML can be used to build simple, but nevertheless useful multimodal interfaces for typical data services for mobile access. Once first implementations of the *voice communicator* and the *multimodal integrator* are available, it should become quite easy for the general web programmer to generate further multimodal interfaces for existing data services.

Yet, returning to the earlier discussion of adaptivity requirements for multimodal dialog systems in section 2.3, we have noticed in our architecture work that some important types of system-controlled adaptation are not supported by the highly modular approach that underlies both VoiceXML and our multimodal architecture. Information about environment characteristics that is easily obtainable in speech recognition, such as the level of ambiance noise and confidence scores in speech recognition, are kept local in the speech recognition component used by the VoiceXML browser (as part of the *implementation platform*, which is not further considered by the standard). They are not accessible in the VoiceXML browser, and therefore neither in the *voice communicator*, nor in the *multimodal integrator*, where they could be used to influence modality selection. Other types of system-controlled adaptation can be implemented in the *multimodal integrator*, provided the interface browsers and interface handlers are designed to provide the required information flows.

Therefore, while we strongly support modular architectures and components that will allow efficient development of multimodal interfaces by a broad base of web developers, we would also appeal to the standards bodies to include dedicated interfaces in the component standards to enable system-controlled, centrally processed adaptivity, for example in the currently planned multimodal extension to VoiceXML.

6. Conclusions

This paper has demonstrated that using VoiceXML, a modular architecture for simple, yet useful multimodal interfaces to data services for mobile access can be defined. We plan to continue this work to define further building blocks for a development model for speech and multimodal interfaces that is streamlined with web technologies. A problem that we have identified is the lack of support for system-controlled adaptation in VoiceXML. We hope that this will be addressed in future work by the standardization bodies.

Acknowledgements
This work was supported within the Austrian competence center program **K***plus*, and by the companies Alcatel, Connect Austria, Kapsch, Mobilkom Austria, and Nokia.

7. References

[1] W3C, “Voice eXtensible Markup Language (VoiceXML) version 1.0,” http://www.w3.org/TR/2000/NOTE-voicexml-20000505/, 2000.

[2] S.L. Oviatt, A. DeAngeli, and K. Kuhn, “Integration and synchronization of input modes during multimodal human-computer interaction,” in *Proc. of CHI97*, New York, 1997, pp. 415–422, ACM Press.

[3] A. Rogozan and P. Delegise, “Adaptive fusion of acoustic and visual sources for automatic speech recognition,” *Speech Communication*, 26, no. 1-2, pp. 149–161, 1998.

[4] C. J. Leggetter and P. C. Woodland, “Speaker adaptation of continuous density HMMs using multivariate linear regression,” in *Proc. of ICSLP 94*, Yokohama, 1994, pp. 451–454.

[5] A. Kobsa and W. Wahlster, Eds., *User Models in Dialog Systems*, Springer, Berlin, 1989.

[6] DISC, “Deliverable d2.7a: State-of-the-art survey of dialogue management tools,” Tech. Rep., Esprit Long-Term Research Concerted Action No. 24823, 1999.

[7] W3C, “Dialog requirements for voice markup languages W3C working draft 23 december 1999,” http://www.w3.org/TR/voice-dialog-reqs/, 1999.

[8] W3C, “Multimodal requirements for voice markup languages W3C working draft 10 july 2000,” http://www.w3.org/TR/multimodal-reqs, 2000.