Towards Mixed-Initiative Concepts in Smart Environments

Dirk Schnelle-Walka
Telecooperation Lab
Darmstadt University of Technology
Darmstadt, Germany
dirk@tk.informatik.tu-darmstadt.de

Josua Arndt
Fakultät für Informationstechnik
University of Applied Science
Mannheim, Germany
josua.arndt1@stud.hs-mannheim.de

Stefan Feldes
Fakultät für Informationstechnik
University of Applied Science
Mannheim, Germany
s.feldes@hs-mannheim.de

ABSTRACT
Automated control of devices in our homes using a PC or wearable touch interfaces is becoming a reality and has already been adopted by industry. Voice has the potential of a more natural interaction but puts high barriers to a deployment due to its error prone and invisible nature. In this paper we introduce a prototype that allows the use of established voice user interface design concepts known from telephony applications in this environment bringing in a conversational approach that is suitable to lower the barriers, leading to a better user experience.

Author Keywords
speech recognition, VoiceXML, JSAPI 2, user experience, error management.

ACM Classification Keywords
H.5.2 Information Interfaces And Presentation: User Interfaces – Voice I/O, Input devices and strategies

INTRODUCTION
Devices that we have in our homes are becoming more and more complex. Beyond the ability to control the devices themselves by their switches, they can also be controlled remotely over the network. Several systems exist that have already been standardized by the industry, e.g. EIB/KNX [1][7, 10]. Behind all that is the user who can simply click on a button on her PC e.g. to toggle the light in the kitchen. User interfaces that can be controlled over mounted touch displays, smart phones or wearable touch interactive devices like the iPad have been adopted by the industry at present. The use of voice for controlling the devices has been considered as an additional modality but was not successful for several reasons:

1. speech recognition is still error prone [2]
2. poor room acoustics and noise make speech recognition even more error prone [8]
3. a special vocabulary has to be learned [23]
4. new devices that are brought into the environment may come with a special vocabulary that conflicts with existing commands [9]
5. the challenge of an ever listening system that can seamlessly detect commands is still a vision [23]
6. there is no standardized programming concept for the development of voice user interfaces in pervasive environments [18]
7. dialogs are distributed [21]

The dominating challenge is the automated speech recognition process that is never perfect enough to produce inputs that are 100% correct. The fact that even humans can not detect speech with that accuracy is mostly ignored [4]. This challenge becomes severer if we think in terms of a moving user who can utter a command at any time at any place. The common way to deal with this issue is the use of headsets, small microphones sewn into the user's clothes or a smart phone that the user picks up, presses a button and is then ready to issue a command [14]. All these workarounds are valid approaches as long as the goal is the possibility to control the environment by voice regardless of the user experience. It can not be assumed that user's will wear headset in a real world scenario. Sewn-in microphones can be possible but are not available of the shelf for reasonable prices right now. The scenario where a user picks up her smart phone decreases the advantages of a hands-and-eyes free access. If the user already presses a button she can also press the correct button to execute the wanted command. A lot of research has been done in the field of equipping the house with room microphones. Although these solutions deliver noteworthy results [2], they are too expensive for a real installation.

The second challenge deals with the ever changing environment. New devices are brought into the environment and

[1] http://www.knx.de

[2] http://www.squarehead.no

CHI 2009, April 3 - 9, 2009, Boston, MA, USA.

others are removed. The dominating strategy to deal with this issue is to use Universal Plug and Play (UPnP) [22]. The concepts rely on a description that each device brings with it and which it broadcasts to the environment using a standard multicast address. Special control points listening to this address can discover these resources with their respective capabilities.

The UPnP device descriptions contain information of the devices and their services. For each service a description of commands, actions, data types and their ranges are specified. This information is used by the control point to generate SOAP messages to control the device. Controlling a device also includes an understanding of the offered methods and capabilities of the devices which can not be covered by the standard. Hence, the semantics of this call remains unclear. Some projects like OwlSpeak [9] try to close this gap using semantic web concepts at the cost of manageability. The creation and maintenance of ontologies is often underestimated. Albertsen et al. state that "Semantic applications are comparable to any other software systems in terms of size and complexity, and thereby require the same rigorous development process as any other system" [1]. In order to really use the ontology in an application, a mapper or translator as described in [19] so that additional effort is needed to keep the changes in the ontology synchronized with the programming model.

Also, there is no commonly agreed way of how to resolve ambiguities of the available devices. Think of a user who wants to turn on his TV by voice. She utters the command "turn on the TV". Due to recognition errors, only the part "turn on" could be recognized. Since this is also part of the commands to turn on the light or the stereo, a clarification dialog is needed to enable the user to achieve her goal without repeating the whole command. The list of possible devices can be determined easily, but there are no convincing concepts available to generate a human-like presentation out of it.

The third challenge is driven from the perspective of the programmer of such environments. Current programming models are driven by research to explore new opportunities and features of such systems. Hence, they generally do not rely on standards that make it easy for developers to reuse their knowledge from the knowledge of comparable systems. We believe that such research systems can only be successful if they are relatively easy to develop. A fact that is ignored by most researchers.

The last challenge deals with the moving user who is not restricted to use a central point of interaction for a single application. Current UI programming concepts are designed for desktop size computers with a local dialog flow control. Dialogs in pervasive environments are distributed by their nature. Jaspis [21] is recognized as *"the most radical approach to architectures for spoken dialog systems"* [12]. It introduces an approach that *"reacts to changes in the information storage rather than on a model of turn-taking"*, using agents to exchange information with the user. But it still relies on client/server paradigms where a single entity controls the dialog flow.

A first steps towards this vision of a standardized API for voice based interaction in pervasive environments has been made by our Mundo Speech API [18]. In this paper we focus on the dominating challenge of handling recognition errors. Based on a categorization of different error types, we show that that current command & control approaches are not suitable to handle them adequately and that a conversational approach has the potential to lead to a better user experience. We also introduce our prototype for a conversational approach of voice based interaction in pervasive environments.

RELATED WORK

This section gives a short overview about existing technologies that is sold today and research conducted in this field.

Gira Speech Control offers the possibility to add command & control capabilities to navigate the menu of their HomeServer[3] and select the actions. Their systems aims to support people with disabilities who can record up to 64 different commands which can be assigned to a menu item or an action. While the users navigate the menu they have to look at the screen and select the next menu item. Once an action is chosen, the HomeServer executes it as a gateway to the KNX/EIB installation. The product also allows the definition of specific scenes to control several items at the same time. For instance, the TV and the stereo can be muted if the user selects the telephony function. It is not possible to utter the whole sequence at once. Moreover, the system is highly speaker dependent which makes it unusable for others.

SemVox is a spin-off of the German Research Center for Artificial Intelligence (DFKI GmbH). They offer a solution for intelligent human-machine-interaction called ODP (Ontology-based Dialog Platform) for the realization of intuitive multimodal applications. The ODP connects various clients by semantic processing of all input such a speech, touch, keyboard or some other modality. Both, the input and output channels of the system are multimodal. The system heavily relies on an ontological representation for each group of clients. ODP claims to support standards such as MRCP/SIP, EMMA and SSML and GUI frameworks such as Ajax, Flash/Flex or JavaFX as well as novel modalities (for example multitouch, gesture recognition or virtual characters) [4]. Applications are developed in a proprietary XML format which is supported by their own workbench [20]. The drawback of this approach is that developers can not reuse their existing knowledge from established standards. Current projects also showed that the use of Ontologies posed a high barrier to deployments.

The EU-funded IST project INSPIRE, developed a "home assistant" [5] hiding the complexity of home appliances through an intuitive interface. It evolved to a framework for multimodal application design, with focus on natural language

[3] http://www.gira.de/produkte/homeserver.html
[4] http://www.semvox.de/en/menu-home.html

control of domestic devices, e.g. TV, video recorder, lamps, blinds, etc. In a collaboration with the Strategic Research Laboratories-Usabilitiy of Deutsche Telekom Laboratories and the DAI-Labor, Berlin University of Technology, the user interface for speech was added to the system to provide remote or local speech interactions. The system consists of the INSPIRE Core controlling the interaction flow, gathering user inputs like spoken utterance, a gesture, or combined expressions and decide how and by which output component to respond or which actions is to be taken. The second part is the HDC (Home Device Controller), developed at the DAI-Labor, which provides access to devices. It defines its own abstract, ontology-based description of devices. Internally. The most interesting component of this system was the additional added user interface for speech utilizing a VoiceXML platform, which only serves as an I/O device while the dialog logic and semantic interpretation of user utterances is done in the INSPIRE Core. This system rediscovered the potential of VoiceXML to be used as a programming language for voice applications apart from telephony environments. VoiceXML documents are generated based on ontological expressions without exploiting the existing knowledge of voice user interface designers that are familiar with this standard. Error management strategies as described above remain untouched in this approach.

Another framework is OwlSpeak [9] that generates VoiceXML documents from ontologies. The current domain and dialog state is reflected using beliefs as starting points for the possible actions that the user may take. An important contribution of this project is that it considers ontologies evolving during a conversation or while the user is following a task. Therefore, they generate VoiceXML snippets with a field containing all the grammars to accept the commands that are possible for the current belief. The fields feature a short timeout which causes the voice browser to throw a noinput event if the user did not say anything within this small timeframe. In the event processing a new VoiceXML dialog is retrieved from the server with updated grammars reflecting the current, possibly changed, beliefs. This approach also allows limited resolution of ambiguities. For now, the user has to take an extra step selecting the name of the relevant ontology out of a list of possible alternatives. Missing in this approach is the incorporation of voice user interface design knowledge. The way how ambiguities are resolved is not very comfortable. Also, the generated VoiceXML snippets lack descent error management strategies.

OVERCOMING THE COMMAND&CONTROL PARADIGM

Voice based interaction relying on the command & control paradigm allows users to accomplish a task without the need to use their hands or to to switch the focus of their attention. Most people working in the field of pervasive computing applications use it as the central means to trigger certain actions. Conversational applications seem to be not a viable substitute in these domains. The user initiative approach of command & control promises that these systems are easy to understand since the user can directly control the action. But there are also drawbacks as Schnelle points out in [16]. The main problem is that users have to learn a special vocabulary. Usually, the system either accepts the utterance or rejects it, without any conversational approach to capture false user input. But the situation is more complex as it is shown in figure 1. The system only does what the user wants if it correctly recognizes the user's utterance. All other cases lead to error situations which need to be corrected. In all cases the error correction is left to the user which was the situation for voice based applications 16 years ago as Kamm depicted in [11]. This is especially the case for falsely accepted input. But both the user and the system should be able to correct errors. Duff established a categorization of different error types [6] that can serve as a basis for error handling. In the following we will have a closer look at these error situations from the perspective of command & control applications. The explanations and examples are taken from [17].

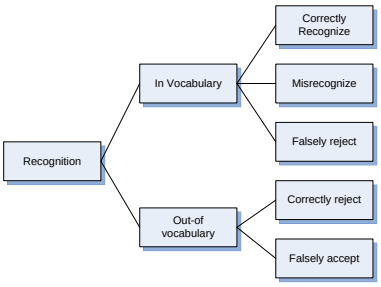

Figure 1. Categorization of Recognition Errors

Level 0 Missing input.
The user was requested to provide an input but did not say anything.
This level is not applicable for command & control since the system does not ask the user to provide some input.

Level 1 Recognition rejection.
The user said something that did not match an expected input.
In these cases today's command & control systems simply reject the command, no actions are taken and the system gives the user another try.

Level 2 Recognizer returns something that cannot be interpreted (makes no sense at all).
This can happen if the user's utterance was falsely mapped to an allowed sequence of words. Imagine a voice automated train schedule where the user says e.g. *The weather will be fine today* which is falsely recognized as *What five today*. Generally this is often referred to as the *out-of-vocabulary problem* (OOV).
Same as for level 1.

Level 3 Recognizer returns something that is not semantically consistent.
In this case an utterance was accepted that matches an allowed sequence of words but has semantic faults. In our train schedule scenario, the user may say something like *I want to leave at 7 kilos*.
Same as for level 1.

Level 4 Recognizer returns semantically well formed, but impossible to fulfill sentences.

Here, the user requests some information that is consistent and makes sense to her, but the system is not able to provide that information. An example would be, if the user asks the train schedule system for a train connection from Darmstadt to Frankfurt, but the connection is currently not served due to ongoing repairs.
Some systems realize that the action is not possible and will thus reject the utterance or provide a short explanation. The system will continue by expecting the next command.

Level 5 Same as 4 with the exception that the impossibility is due to the dialog context.
An example for this level is that the user selects the fifth available train connection, although there are only four of it.
Since there is no dialog context, this is the same as level 4.

Level 6 The back-end system fails to fulfill the command
If e.g. the database connection is lost it will not be possible to deliver any information. Current systems are able to read the success of the triggered action over the used communication channel, e.g. KNX/EIB, and report error messages back to the user. Hence, this is also comparable to level 4.

Level 7 User initiated error correction.
This level summarizes all kinds of errors that are not covered by the levels above and can not be identified automatically.
Falsely accepted commands lead to this situation where typical command & control systems expect that it is easy for the user to undo the command.

Summing up, there are only three known behaviours how to react to error:

1. let the user reenter the command
2. explain the error situation and start over
3. let the user correct the error

The user experience in these situations is that

1. the user keeps on entering the command until she becomes tired of it
2. the system is not reliable
3. the system does not do what the user wants

The categorization of potential causes for an error and the different levels of error situations remain unexploited. The experience from telephony based applications shows that efficient error handling can increase the success for applications with even poor recognition accuracy dramatically. In [15] the authors show that the task completion rate increased from 86.4% to 93.4% and the average number of turns reduced by three after a better error handling method had been installed. The drawback of poorly constructed error handling is that it may bring unwanted complexity to the system and cause new errors and annoyances.

Another benefit of a conversational approach to command & control systems is that it can be used for clarification issues as discussed above. Conversational approaches are usually implemented as mixed initiative. Here, the user can take the initiative an if some data is missing, the application will ask for the missing data, imitating a natural human-to-human conversation. VoiceXML [13] claims to support this dialog strategy. In VoiceXML, mixed-initiative dialogs are realized as slots that are filled by a grammar containing SISR (Semantic Interpretation for Speech Recognition)[5] tags. The SRGS (Speech Recognition Grammar Specification)[6] processing will create a JavaScript object from these tags whose attributes are then mapped to slots defined in the VoiceXML dialog. If the user does not provide the entire command, the voice browser continues with the slots that remain unfilled. *Real* mixed initiative dialogs with information overloading and out-of-focus answers will not be possible using this technologoy, but many applications based on VoiceXML have been successfully developed and deployed. Efficient dialogs coming close to a natural interaction are only possible with carefully designed dialogs. Many guidelines, like the ones given in [3] and [2] exist and there are also first attempts to transfer the idea of design patterns [16] to the design of voice user interfaces.

DEMO IMPLEMENTATION

The system we developed is deployed in a house with a EIB/KNX installation. A Gira HomeServer controls the lights, shutters, air-condition, heating installation, TV and Radio and a IP-Gateway as shown in figure 2. We believe that the

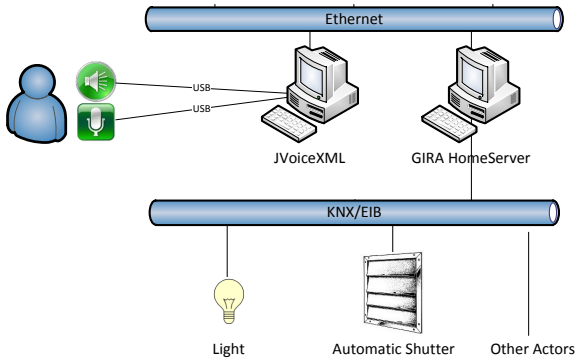

Figure 2. Architecture of the prototype

concepts introduced in this paper are also applicable to interact with smart objects although we target a room control. The setup offers access to the sensors and actors through the network with the help of so-called IP-telegrams. A PC

[5] http://www.w3.org/TR/semantic-interpretation/
[6] http://www.w3.org/TR/speech-grammar/

with the JVoiceXML[7] Server enables speech access. For sound in- and output a Yamaha PJP-10UR was used. It is a web conference microphone speaker featuring a high performance echo canceller. Efficient echo cancellation is necessary to avoid outputs to trigger the system. We observed good recognition accuracy even at a distance of more than three meters. For the speech recognition and speech synthesis, we used the Windows 7 built-in engines. They were accessed over our own implementation of JSAPI 2 platform with a bridge to Windows SAPI. This implementation including the bridge is available as an open source component[8]. For now, we are using a push-to-talk mechanism to activate the main dialog by initiating a call to the VoiceXML server. JVoiceXML allows the emulation of a call through the speaker and the microphone of the PC. The dialog is designed to collect all needed information for a full qualified command with the help of mixed initiative concepts. This means that it is only necessary to understand only a part of the whole command. If some information is missing, the system will take the initiative and ask for that data. The use of dedicated error management strategies, like rapid reprompt (cf. [16]) helps the user to enter the command and all needed parameters.

After a command is issued successfully, the corresponding IP-Telegram is sent to the GIRA HomeServer. The server evaluates it and activates the addressed actor over the KNX/EIB bus. After the action has been initiated, the processing continues with expecting the next command.

A typical dialog is shown below:

User: Please close the shutter
System: Which shutter shall be closed?
User does not know how to continue and says nothing
System: Do you want to close the shutter to the garden or to the terrace?
User: The terrace

With the help of the system, the user was able to complete the command without any learning effort. After the action is initiated, the user may utter enter the next command, e.g. to stop the closing action. With our approach, we rely on standards that make it possible to reuse existing design knowledge of domain experts.

CONVERSATIONAL APPROACH

The analysis from above showed that the error levels 0 and 4-6 cannot be addressed by a conversational error management strategy. The error levels 1 to 3 and 7 are currently handled by the same mechanisms but which can be treated differently as described in our pattern language for error management in voice user interfaces [17]. From this pattern language, the following patterns are suitable means to recover from errors of the levels 1 and 2: ESCALATING DETAIL and RAPID REPROMPT. They provide an error recovery strategy when the speech recognizer does not return a recognition hypothesis and guides the user in an error situation with the right amount of information. This is in particular the case for level 1 and 2. A dialog could look as follows

User: Please close the shutter
User: *unrecognized input*
System: I did not understand you. Please repeat.
User: *unrecognized input*
System: I did not understand you. You can simply name the device to control and what to do with it.
User: *unrecognized input*
System: Sorry, I still did not understand you. Which device do you want to control?
User: The shutter.
System: Do you want to close the shutter?
User: Yes

The system first gives the user another try to enter her command. If this still fails, the system increases the level of detail and finally directs the user to enter the relevant parts of the command.

If only part of the utterance could be mapped successfully to a command as it is the case for level 3 errors, SELECTION FROM A LIST and IMPLICIT CONFIRMATION can be applied in addition. An example is given in the previous section where the user selects the wanted device from a list. The recognition of the device is implicitly confirmed in the system response for clarification.

For level 7 errors e.g. the patterns GLOBAL ERROR CORRECTION, THREE TIERED CONFIDENCE, IMMEDIATE FEEDBACK, EXPLICIT CONFIRMATION and IMPLICIT CONFIRMATION could be applied. GLOBAL ERROR CORRECTION could be used in the worst case to offer alternative ways to control the devices if the recognition fails at all. THREE TIERED CONFIDENCE could be used if the recognizer returns with a recognition hypothesis with a low confidence score. In these cases, the system imitates the behaviour of humans in a dialog asking for confirmation if there was some uncertainty about the heard words by simply asking to repeat the missed part. However, this has to be verified in user studies that we are about to conduct. This is especially true for the use of EXPLICIT CONFIRMATION which can be applied to retrieve the user's confirmation before executing the action. If the command can be simply undone this may hinder users from using voice based control at all since dialogs become too rigid and lengthy.

SUMMARY AND OUTLOOK

In this paper we described our prototype for a VoiceXML based dialog control that can be used to develop applications in pervasive environments. The mixed initiative concepts are a suitable means to adapt established error correction strategies that are known from the domain of telephony applications. Moreover, it seems to be a promising approach to overcome the pitfalls of typical command & control systems where the user has to learn a special vocabulary before she is able to use the system. We introduced some first

[7] http://jvoicexml.sourceforge.net
[8] http://jsapi.sourceforge.net

thoughts how our pattern language for voice user interface design [16] could be used towards conversational user interfaces, but which have to be verified in user studies.

The use of JVoiceXML together with the JSAPI 2 allows us to make use of our Mundo Speech API [18] that can be hooked as an implementation platform. There is ongoing work to investigate the suitability of this approach for nomadic users. Another planned enhancement is to adopt OwlSpeak [9] to create VoiceXML documents on the fly. In order to incorporate voice user interface design knowledge, we plan to enhance the generation of VoiceXML scripts to use our pattern language.

Acknowledgements
Thanks to Bernd Klein from Cibek[9] who let us use his EIB/KNX equipped house and for the good atmosphere.

REFERENCES

1. T. Albertsen and E. Blomqvist. Describing ontology applications. In *ESWC '07: Proceedings of the 4th European conference on The Semantic Web*, pages 549–563, Berlin, Heidelberg, 2007. Springer-Verlag.
2. B. Ballentine. *It's Better to Be a Good Machine Than a Bad Person: Speech Recognition and Other Exotic User Interfaces at the Twilight of the Jetsonian Age*. ICMI Press, Mar. 2007.
3. M. H. Cohen, J. P. Giangola, and J. Balogh. *Voice User Interface Design*. Addison-Wesley, Boston, jan 2004.
4. N. Deshmukh, R. J. Duncan, A. Ganapathiraju, and J. Picone. Benchmarking human performance for continuous speech recognition. In *REFERENCES Page 23 RFC 1319 B. Kaliski. RFC 1319: The MD2 Message-Digest Algorithm*, pages 2486–2489, 1996.
5. T. Dimopulos, S. Albayrak, K. Engelbrecht, G. Lehmann, and S. Moller. Enhancing the Flexibility of a Multimodal Smart Home Environment. *Fortschritte der Akustik*, 33(2):639, 2007.
6. D. Duff, B. Gates, and S. LuperFoy. An architecture for spoken dialogue management. In *Proceedings of the 1996 International Conference on Speech and Language Processing (ICSLP)*, 1996.
7. EN50090-5-1:2005: Home and building electronic systems (HBES). media and media dependent layers. power line for HBES class 1, Mar. 2005.
8. D. Gelbart and N. Morgan. Double the trouble: Handling noise and reverberation in far-field automatic speech recognition, 2002.
9. T. Heinroth, D. Denich, and A. Schmitt. OwlSpeak - adaptive spoken dialogue within intelligent environments. In *8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 666 – 671, Mar. 2010.

[9] `http://www.cibek.de`

10. ISO/IEC 14543: Information technology – home electronic system (HES) architecture, 2010.
11. C. Kamm. *Voice Communication Between Humans and Machines*, chapter User Interfaces for Voice Applications, pages 422–442. National Academy Press, Washington D.C., 1994.
12. M. McTear. New directions in spoken dialogue technology for pervasive interfaces. In *Proceedings of the Workshop on Robust and Adaptive information Processing for Mobile Speech interfaces*, pages 57–64, 2004.
13. M. Oshry, R. Auburn, P. Baggia, M. Bodell, D. Burke, D. C. Burnett, E. Candell, J. Carter, S. McGlashan, A. Lee, B. Porter, and K. Rehor. Voice Extensible Markup Language (VoiceXML) Version 2.1, W3C Recommendation. `http://www.w3.org/TR/voicexml21/`, June 2007.
14. S. Ott, W. Spiegl, S. Soutschek, A. Maier, S. Steidl, and E. Nöth. Home Assistance System for Elderly People. In R. B. C. on Bio-Medical Engineering Communication, editor, *Proceedings of the 5th Russian-Bavarian Conference on Biomedical Engineering*, 2009.
15. H. Sagawa, T. Mitamura, and E. Nyberg. Correction grammars for error handling in a speech dialog system. In *HLT-NAACL '04: Proceedings of HLT-NAACL 2004: Short Papers on XX*, pages 61–64, Morristown, NJ, USA, 2004. Association for Computational Linguistics.
16. D. Schnelle. *Context Aware Voice User Interfaces for Workflow Support*. PhD thesis, TU Darmstadt, 2007.
17. D. Schnelle-Walka. A Pattern Language for Error Management in Voice User Interfaces. In *Proceedings of EuroPLoP 2010*, July 2010.
18. D. Schnelle-Walka and S. Radomski. An api for voice user interfaces in pervasive environments. In *SiMPE 2010 Conference Proceedings*, Sept. 2010.
19. J. Serrano, J. Serrat, and A. Galis. Ontology-based context information modelling for managing pervasive applications. In *International Conference on Autonomic and Autonomous Systems, 2006.*, pages 47–50, Silicon Valley, CA, 2006.
20. D. Sonntag, G. Sonnenberg, R. Nesselrath, and G. Herzog. Supporting a rapid dialogue engineering process. In *Proceedings of the First International Workshop On Spoken Dialogue Systems Technology (IWSDS)*, 2009.
21. M. Turunen, J. Hakulinen, K.-J. Räihä, E.-P. Salonen, A. Kainulainen, and P. Prusi. Jaspis an architecture and applications for speech-based accessibility systems. *IBM Systems Journal*, 44(3):485–504, 2005.
22. UPnP Forum. UPnP Device Architecture 1.1, Oct 2008.
23. G. Weinberg and B. Harsham. Object-oriented multimodality for safer in-vehicle interfaces. In *SiMPE: 5th workshop on speech in mobile and pervasive environments*, 2010.