AdaRTE: An Extensible and Adaptable Architecture for Dialog Systems

L.M. Rojas-Barahona
Dip. Informatica e Sistemistica
Università di Pavia
Pavia, 27100, IT
linamaria.rojas@unipv.it

T. Giorgino
Dip. Informatica e Sistemistica
Università di Pavia
Pavia, 27100, IT
toni.giorgino@unipv.it

Abstract

Dialog Systems have been proven useful to provide the general public with access to services via speech devices. In this paper, we present AdaRTE, an Adaptable Dialog Architecture and Runtime Engine. AdaRTE uses dynamic Augmented Transition Networks and enables the generation of different backend formats; for instance, it supports VoiceXML generation to guarantee portability and standards compliance. The scope of AdaRTE is to provide a ground for deploying complex adaptable dialogs such as those found in the patient-care domain, and for experimenting with innovative speech solutions including Natural Language Processing. AdaRTE is an extensive architecture for dialog representation and interpretation, which helps developers to layout dialog interactions through a high level formalism whilst allowing the inclusion of voice applications best-practices.

1 Introduction

Dialog technologies have been widely applied in different domains. Previous to Voice Browsers (VB), proprietary technology was the response to vocal applications deployment. The speech systems adopted either custom code, or proprietary dialog-manager based solutions. Linear script, state transition networks and plan-based were among the available technologies for dialog management systems. Generally, the deployment of any of these techniques requires heavily scripted solutions. On the other hand, the multitude of dialog technology vendors naturally resulted in a proliferation of incompatible languages across vendors and platforms.

More recently, the advent of VoiceXML allows to deploy dialog systems in a Web-based environment (McGlashan et al., 2004). Its delivery contributed to reduce the proliferation of incompatible dialog formalisms by offering one standard for voice applications. However, VoiceXML has inherent limitations which are well analyzed in (Mittendorfer et al., 2002), such as its declarative and static structure, difficulty accessing remote resources (databases and ontologies) and lack of means for efficient and heavy computation. Furthermore, VoiceXML does not allow an explicit visualization of the dialog flow because of its form-filling mechanism and, like web based technologies, has to be generated by other code dynamically.

Perhaps the strongest limit pointed out by the research community is that VoiceXML does not directly support neither dynamic natural language understanding and generation, nor multimodality. As a consequence, extensions to VoiceXML has been proposed in literature: DialogXML was applied to car telematics services; in this approach, the VB was extended to support NLP KANTOO generated grammars (Hataoka et al., 2004). Other VoiceXML-generative approaches are presented in (Hamerich et al., 2004) which follows a database-oriented approach, and (Di Fabbrizio and Lewis, 2004) which is seemingly targeted towards customer care tasks

Bridging the Gap: Academic and Industrial Research in Dialog Technologies Workshop Proceedings, pages 64–67,
NAACL-HLT, Rochester, NY, April 2007.

with sophisticated call routing, rather than the structured enquiry data collection tasks found in chronic patient management. We believe that a big effort should still be done in adapting dialog systems best practices (Balentine and Morgan, 2001), such as confirmation strategy, adaptability, mixed initiative, usable speech interfaces for users and graphical interfaces for developers in VoiceXML-based frameworks. Commonly, the process of deploying dialog systems was complicated, costly, time demanding and required speech technology experts. A leaner development methodology is particularly necessary when considering domains in which available resources for development are limited; such a case is that of the health care domain, in which voice applications have been used for several home-care interventions successfully. (Young et al., 2001; Giorgino et al., 2004; Bickmore et al., 2006).

In this paper, we present an architecture devised to overcome all these issues. Features were thought to reduce dialog system development effort through reuse, support for hierarchical network specification, adaptable decision takers and best practices adoption. We built AdaRTE, which implements these features for dialog deployment, and we present results obtained through the partial prototyping of two telephony-linked systems: the first inspired by the Chronic Obstructive Pulmonary Disease (COPD) care (Young et al., 2001), and the second by the Homey dialog system for hypertensive patient home management (Giorgino et al., 2004). Our effort was mainly focused on health care dialogs systems, since our solution is targeted at offering a standards-compliant way of deploying dialog systems, whose additional peculiarities are extensibility, support for complex dialog flows and low-cost development.

2 AdaRTE Architecture

The architecture we propose (figure 1) is primarily composed of a dialog interpreter, a runtime engine and an interface media realizer for backends generation. A running system interacts with users which can be grouped in three main role categories: Application developers, patients and case managers, i.e. case manager nurses.

The dialog flow and structure are represented in a well-defined XML formalism (XML dialog description). To cooperate with standards-based speech recognition software and respond to telephone-originated events, AdaRTE acts as a web server, generating VoiceXML or HTML code dynamically. Prompts, questions and other elements are the nodes (here named blocks) of an Augmented Transition Network (ATN) that specifies the flow of the conversation. Blocks are represented in the description by XML tags. When the system is started, the XML dialog description is read by AdaRTE which maintains an internal representation of the dialog, and executes it when a call comes in. Consequently, it activates the dialog blocks in sequence or according to a specific criterion, constructs prompts, interprets the answers returned by the caller through the voice platform, and interacts with external resources as appropriate.

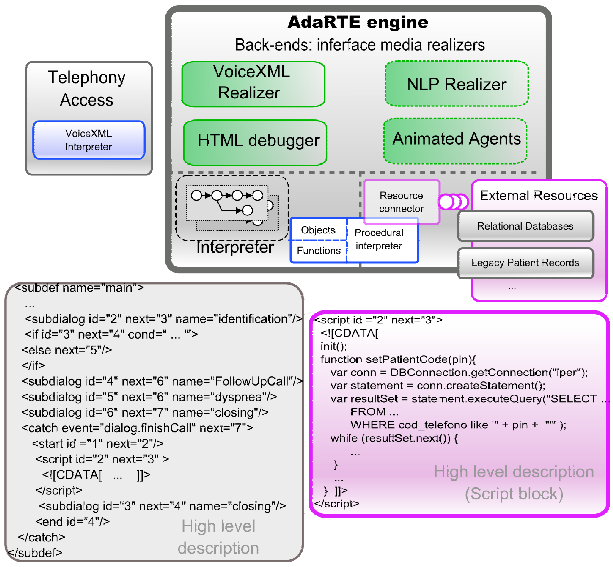

Figure 1: AdaRTE architecture block diagram

Usually, an ATN is associated with a context and, here, we call this structure subdialog. When a call is setup, the main subdialog is retrieved and started; in its turn, it can invoke other subdialogs, and so forth. If the execution flow reaches the end of the main subdialog, the call is terminated. Subdialogs can also terminate unexpectedly if an exception occurs, and, in this case, an exception handler is executed.

In addition, we grant the application of *best-practices* through the configuration of thresholds and n-best lists confirmation strategy related to ques-

tions. Adaptability i.e. flexibility according to users experience with the system, is reached by using *containers*. They are used for common tasks, in which one of several subdialogs are selected according to a specified policy such as randomly, in sequence, ordered by call number, according to any externally defined schedule or a criterion based on reinforcement learning techniques. Inclusion of procedural code at user-level is essential for flexibility, interoperability, and ease of programming. AdaRTE allows to embed snippets of code, which are written in the ECMAScript standard language, into script blocks. These user-written code is run in a separate execution environment with extensive facilities and standard libraries. This also enables access to external resources, including databases, ontologies, or any other commodity library, i.e a probabilistic-based library.

Currently, semantic recognition is implemented either inside the engine itself or leveraging the context-free grammar (CFG) formats offered by the VB. However, we strongly believe in the necessity of integrating a more elaborated semantic recognition solution by supporting NLP and more expressive grammars. In addition, since the architecture of AdaRTE is extensible, it would be possible to integrate any other backend. For instance, we foresee the adoption of multimodality through the generation of an enriched markup language which would be understood by an external animation generator module. Work is in progress toward these directions.

Differing from other VoiceXML-generative frameworks, AdaRTE is oriented to the medical domain, which requires adaptable dialogs with complex structures. Also, it offers a new level of flexibility to developers by allowing external resources access inside script blocks. Moreover, the high level dialog description is intuitive, thus simple dialogs could be implemented by not expert authors. Finally, AdaRTE was thought to be a standard-compliant architecture for incremental adoption of voice formalisms, i.e. lexicalized grammar-based NLPs.

3 Results

Currently, the AdaRTE framework is operative. It has been beta-tested with two realistic health care dialog systems, derived by actual systems deployed and validated in the previous years. The first one is based on a prototype based on the TLC-COPD dialog deployed by the Boston MISU group and others (Young et al., 2001). For this specific example, we used Tellme Studio[1] as VSP. This pilot's deployment demanded less than two weeks of man effort. The fulfilled activities were database schema definition and data preparation together with the dialog deployment. This dialog is executed in English language and uses keypad touch-tone (DTMF) interaction.

The second test case is the partial reimplementation of the Homey dialog system. Homey had been deployed for the management of hypertensive patients (Giorgino et al., 2004). The system included an extensive Electronic Health Records system with storage of personal data and profiles, in order to support dialog adaptability. Reengineering part of the Homey proprietary dialog manager to the AdaRTE architecture took approximately three weeks. The development of this prototype involved the following activities: VSP evaluation, database definition and grammars and dialog deployment. Unlike the TLC-COPD pilot, this system uses speech rather than DTMF input. We built grammars by using the Nuance GSL language and SRGS grammar formats. The language of the dialog is Italian and the dialog was deployed by using Voxpilot as VSP[2]. The expressiveness of the dialog formalism yielded an important reduction of time invested in developing both prototypes whilst facilitating component reuse in each dialog.

4 Future enhancements

A large body of research on the optimization of spoken interfaces is available (Walker et al., 1997). Some of the results of the research have been condensed into best practices (Balentine and Morgan, 2001). For example, more complex confirmation strategies with respect to simple "yes/no" answers should be adopted. Inclusion of such techniques into custom-developed systems is complex. A big advantage in using the interpretable and high-level dialog representation language proposed in this work is that

[1] Tellme Studio. https://studio.tellme.com/

[2] VoxBuilder. http://www.voxbuilder.com/

such "dialog practices" can be incorporated seamlessly into the underlying dialog interpretation logic, removing the burden from the dialog developer.

Currently, we have a strong commitment on the integration of a more elaborated semantic interpretation mechanism by integrating AdaRTE with a NLP application that supports more expressive grammars. In this way, not only recognition does not depend on the grammars supported by VBs, but also more natural dialogs will be supported, so patients perception of the dialogs will improve. In addition, a multimodal extension of AdaRTE through the implementation of a facial expressions and gestures realizer should be considered for future research as well as the extension of automated discourse planning facilities.

5 Conclusion

We have presented an architecture for next-generation dialog representation and interpretation and built an engine for dialog deployment. AdaRTE supports high-level dialog representations whilst implicity takes care of aspects and best practices that are non considered in the current voice and multimodal standards i.e VoiceXML. It supports VoiceXML to communicate to VBs as one of the interpretation and generation backends. We have reengineered two health-care dialog prototypes, chosen as real world test cases, by using the novel architecture and showed that dialog development time is remarkably optimized with respect to customized coding.

The AdaRTE system is motivated not only as a reliable platform for dialog deployment, but also as a framework for incorporating advanced features of speech recognizers, including increased support to adaptability, natural language understanding and generation, and multimodality.

Acknowledgments

The authors thank Dr Robert Friedman (Medical Information Systems Unit, Boston, MA) for information on the TLC-COPD dialogs, and Timothy Bickmore (College of Computer and Information Science, Northeastern University, Boston, MA) for the discussions that led to the AdaRTE project.

References

Bruce Balentine and David P. Morgan. 2001. *How to Build a Speech Recognition Application. A Style Guide for Telephony Dialogues*, Second ed. EIG Press, San Ramon, CA.

Timothy Bickmore, Toni Giorgino, Nancy Green, Rosalind Picard 2006. Special Issue on Dialog Systems for Health Commu-nication. *Journal of Biomedical Informatics Elsevier*,39:465–467.

Giuseppe Di Fabbrizio and Charles Lewis. 2004. Florence: a Dialogue Manager Framework for Spoken Dialogue Systems. *In: Proceedings of the 8th International Conference on Spoken Language Processing (ICSLP 2004)*,Jeju, Jeju Island, Korea.

Toni Giorgino, Ivano Azzini, Carla Rognoni, Silvana Quaglini, Mario Stefanelli, Roberto Gretter and Daniele Falavigna. 2004. Automated spoken dialogue system for hyper-tensive patient home management. *International Journal of Medical Informatics. Elsevier*,74:159–167.

Hamerich Stefan, Schubert Volker, Schless Volker, de Cordoba Ricardo, Pardo Jose d'Haro Luis, Kladis Basilis, Kocsis Otilia and Igel Stefan. 2004. Semi-Automatic Generation of Dialogue Applications in the GEMINI Project. *Sigdial*.

Nobuo Hataoka, Yasunari Obuchi, Teruko Mitamura and Eric Nyberg. 2004. Robust speech dialog interface for car telematics service. *ieeecnc*,331–335.

Scott McGlashan, Burnett Daniel C, Carter Jerry, Danielsen Peter, Ferrans Jim, Hunt Andrew, Bruce Lucas, Porter Brad, Rehor Ken, Tryphonas Steph . 2004. Voice Extensible Markup Language (VoiceXML) Version 2.0. http://www.w3.org/TR/voicexml20/.

Markus Mittendorfer, Georg Niklfeld, Werner Winiwarter 2002. Making the voice web smarter-integrating intelligent component technologies and VoiceXML. *In: Proceedings of the Second International Conference on Web Information Systems Engineering*, 2:126-131.

Marilyn A. Walker, Diane J. Litman, Candace A. Kamm and Alicia Abella. 1997. *PARADISE: A Framework for Evaluating Spoken Dialogue Agents. In: Proceedings of ACL/EACL.*,271-280.

Melissa Young, David Sparrow, Daniel Gottlieb, Alfredo Selim, Robert H. Friedman. 2001. A telephone-linked computer system for COPD care. *Chest*,119:1565–1575.