Griol D., Molina J.M. (2014) A Framework to Develop Adaptive Multimodal Dialog Systems for Android-Based Mobile Devices. In: Polycarpou M., de Carvalho A.C.P.L.F., Pan JS., Woźniak M., Quintian H., Corchado E. (eds) Hybrid Artificial Intelligence Systems. HAIS 2014. Lecture Notes in Computer Science, vol 8480. Springer, Cham

DOI: 10.1007/978-3-319-07617-1_3

A Framework to Develop Adaptive Multimodal Dialog Systems for Android-Based Mobile Devices*

David Griol and José Manuel Molina

Applied Artificial Intelligence Group
Computer Science Department
Carlos III University of Madrid
Avda. de la Universidad, 30, 28911 - Leganés, Spain
{david.griol,josemanuel.molina}@uc3m.es

Abstract. Mobile devices programming has emerged as a new trend in software development. The main developers of operating systems for such devices have provided APIs for developers to implement their own applications, including different solutions for developing voice control. Android, the most popular alternative among developers, offers libraries to build interfaces including different resources for graphical layouts as well as speech recognition and text-to-speech synthesis. Despite the usefulness of such classes, there are no strategies defined for multimodal interface development for Android systems, and developers create ad-hoc solutions that make apps costly to implement and difficult to compare and maintain. In this paper we propose a framework to facilitate the software engineering life cycle for multimodal interfaces in Android. Our proposal integrates the facilities of the Android API in a modular architecture that emphasizes interaction management and context-awareness to build sophisticated, robust and maintainable applications.

Keywords: Dialog systems, Multimodal interaction, Android, Mobile devices, User adaptation, Statistical methodologies.

1 Introduction

Continuous advances in the development of information technologies have currently led to the possibility of accessing information and services on the Internet from anywhere, at anytime and almost instantaneously. In addition, these technological advances have made possible the creation of powerful mobile devices capable of running network applications and accessing web services and information through wireless connections. Smartphones and tablets are widely used today to access the web, but mainly through web browsers or graphical user interfaces.

*This work was supported in part by Projects MINECO TEC2012-37832-C02-01, CICYT TEC2011-28626-C02-02, CAM CONTEXTS (S2009/TIC-1485).

Different technologies have recently emerged to facilitate the accessibility of these devices, which reduced size makes them difficult to operate in some situations and specially for some user groups. For example, multimodal dialog systems [1] can be employed to build more natural interaction with mobile devices by means of speech. They can be defined as computer programs designed to emulate communication capabilities of a human being including several communication modalities, such as speech, tactile and visual interaction.

In addition, these systems typically employ several output modalities to interact with the user, which allows to stimulate several of his senses simultaneously, and thus enhance the understanding of the messages generated by the system. This is particularly useful for people with visual or motor disabilities, allowing their integration and the elimination of barriers to Internet access [2]. For this reason, multimodal conversational agents are becoming a strong alternative to traditional graphical interfaces which might not be appropriate for all users and/or applications [1].

In this paper, we propose a domain-independent framework to develop multimodal dialog systems for mobile devices. Currently the 75% of smartphones and tablets operate with the Android OS [3]. Also, there is an active community of developers who use the Android Open Source Project and have made possible to have more than one million applications currently available at the official Play Store, many of them completely free. For these reasons, our framework makes use of different facilities integrated in Android-based devices.

The remainder of the paper is as follows. Section 2 briefly describes the motivation of our proposal and related work. Section 3 describes the proposed framework to develop adaptive multimodal dialogs systems for mobile devices. Section 4 presents the application of our proposal to developed an advanced multimodal city street guide for Android-based mobile devices. This section also presents the results of a preliminary evaluation of this system. Finally, Section 6 presents some conclusions and future research lines.

2 State of the Art

Although there are currently different approaches to make web contents available using multimodal interaction, they present important limitations. Some of them add a vocal interface to an existing web browser [4]. Others are focused on specific tasks, as e-commerce [5], chat functionalities [6], database access [7], etc. Finally, the solution could be restricted to access information of a limited domain, like in [8], where the dialog system works for selected on-line resources. Several traditional information retrieval systems have been also extended with a vocal interface. However, these applications usually emphasize on the search of documents and not on the interaction with the user.

Additionally, several studies have reported that providing applications with multimodal interfaces is becoming a way to achieve more efficient, pleasant and adapted interaction for mobile applications [9]. In human conversation, speakers adapt their message and the way they convey it to their interlocutors and to the

context in which the dialog takes place. This way, information related to the environment and users presence and location is essential to achieve this adaptation. Recent portable devices (e.g. Android-based mobile devices) are equipped with a diversity of input and output technologies and sensors (accelerometers, multitouch screens, compasses) and start using these to support some form of very basic multimodal interaction. They can be employed to adapt the operation of the multimodal system by taking into account both the context of the interaction and user's specific preferences and previous interactions with the system.

Our proposed framework allows an advance in this direction by considering these valuable sources of contextual information for the development of adaptive multimodal interfaces [10]. To do this, a statistical methodology is proposed to flexible adapt the operation of the system taking into account both user's specific interactions and preferences and also environmental conditions.

3 Proposed Framework to Develop Adaptive Multimodal Dialog Systems for Android-Based Mobile Devices

Figure 1 shows the proposed framework. A spoken dialog system integrates five main tasks to deal with user's spoken utterances: automatic speech recognition (ASR), natural language understanding (NLU), dialog management (DM), natural language generation (NLG), and text-to-speech synthesis (TTS).

Speech recognition is the process of obtaining the text string corresponding to an acoustic input. Our proposal integrates the Google Speech API to include the speech recognition functionality in a multimodal dialog system. Speech recognition services have been available on Android devices since Android 2.1 (API level 7). The ASR functionality is available by means of a microphone icon on the Android keyboard, which activates the Google speech recognition service. Language-configurable messages can be predefined for specific events like no speech detected, no suitable match for the user's utterance, or no Internet connection is available.

Using the Google Speech API (package *android.speech*), speech recognition can be carried out by means on a *RecognizerIntent*, or by creating an instance of *SpeechRecognizer*. The former starts the intent and process its results to complete the recognition, providing feedback to the user to inform that the ASR is ready or there were errors during the recognition process. The latter provides developers with different notifications of recognition related events, thus allowing a more fine-grained processing of the speech recognition process. In both cases, the results are presented in the form of an N-best list with confidence scores.

Once the conversational agent has recognized what the user uttered, it is necessary to understand what he said. Natural language processing generally involves morphological, lexical, syntactical, semantic, discourse and pragmatical knowledge. Lexical and morphological knowledge allow dividing the words in their constituents distinguishing lexemes and morphemes. We propose the use of grammars in order to carry the semantic interpretation of the user inputs.

As explained in the introduction section, a multimodal dialog system involves user inputs through two or more combined modes, which usually complement

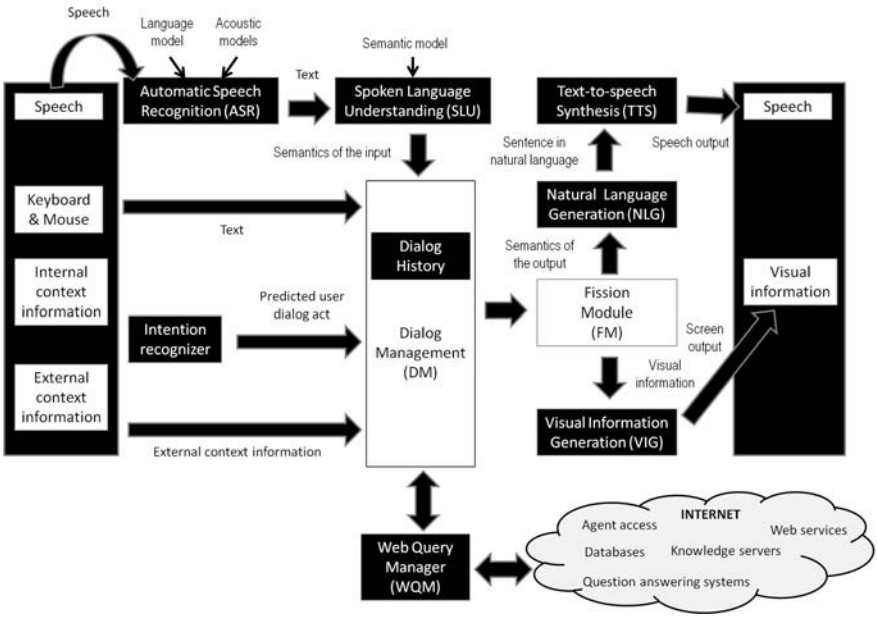

Fig. 1. Proposed framework for the generation of multimodal dialog systems in Android-based mobile devices

spoken interaction by also adding the possibility of textual and tactile inputs provided using physical or virtual keyboards and the screen. In our contribution, we want also to model the context of the interaction as an additional valuable information source to be considered in the fusion process.

We propose the acquisition of external context by means of the use of sensors currently supported by Android devices. Android allows applications to access location services using the classes in the *android.location* package. The central component of the location framework is the *LocationManager* system service, also the Google Maps Android API permits to add maps to the application, which are based on Google Maps data. This API automatically handles access to Google Maps servers, data downloading, map display, and touch gestures on the map. The API can also be used to add markers, polygons and overlays, and to change the user's view of a particular map area. To integrate this API into an application, is it required to install the Google Play services libraries.

Most Android-powered devices have built-in sensors that measure motion, orientation, and various environmental conditions. These sensors are capable of providing raw data with high precision and accuracy, and are useful to monitor three-dimensional device movement or positioning, or monitor changes in the ambient environment near a device. The Android platform supports three main categories of sensors. Motion sensors measure acceleration forces and rotational forces along three axes. This category includes accelerometers, gravity sensors,

gyroscopes, and rotational vector sensors. Environmental sensors measure various environmental parameters, such as ambient air temperature and pressure, illumination, and humidity. This category includes barometers, photometers, and thermometers. Finally, position sensors measure the physical position of a device. This category includes orientation sensors and magnetometers.

The Android sensor framework (*android.hardware* package) allows to access these sensors and acquire raw sensor data. Some of these sensors are hardware-based and some are software-based. Hardware-based derive their data by directly measuring specific environmental properties, such as acceleration, geomagnetic field strength, or angular change. Software-based sensors derive their data from one or more of the hardware-based sensors (e.g., linear acceleration and gravity sensors).

Android also provides several sensors to monitor the motion of a device. Two of these sensors are always hardware-based (the accelerometer and gyroscope), and three of these sensors can be either hardware-based or software-based (the gravity, linear acceleration, and rotation vector sensors). Motion sensors are useful for monitoring device movement, such as tilt, shake, rotation, or swing. All of the motion sensors return multi-dimensional arrays of sensor values for each SensorEvent. Two additional sensors allow to determine the position of a device: the geomagnetic field sensor and the orientation sensor. The Android platform also provides a sensor to determine how close the face of a device is to an object (known as the proximity sensor). The geomagnetic field sensor and the proximity sensor are hardware-based. The orientation sensor is software-based and derives its data from the accelerometer and the geomagnetic field sensor.

Finally, four sensors allow monitoring various environmental properties: relative ambient humidity, light, ambient pressure, and ambient temperature near an Android-powered device. All four environment sensors are hardware-based and are available only if a device manufacturer has built them into a device. With the exception of the light sensor, which most device manufacturers use to control screen brightness, environment sensors are not always available on devices. Unlike most motion sensors and position sensors, environment sensors return a single sensor value for each data event.

Regarding internal context, our proposal is based on the traditional view of the dialog act theory, in which communicative acts are defined as intentions or goals. Our technique is based on a statistical model to predict user's intention during the dialog, which is automatically learned from a dialog corpus. This model is used by the system to anticipate the user's needs by dynamically adopting their goals and also providing them with unsolicited comments and suggestions, as well as responding immediately to interruptions and provide clarification questions. The model takes into account the complete history of the interaction and also the information stored in user profiles.

The dialog manager of the system has to deal with different sources of information such as the NLU results, database queries results, application domain knowledge, and knowledge about the users and the previous dialog history to select the next system action. We propose a statistical methodology that combines

multimodal fusion and dialog management functionalities. To do this, a data structure is introduced to store the information provided by the user's inputs, the user's intention model, and the context of the interaction.

The modality fission module receives abstract, modality independent presentation goals from the dialog manager. The multimodal output depends on several constraints for the specific domain of the system, e.g., the current scenario, the display size, and user preferences like the currently applicable modality mix. This module applies presentation strategies that decompose the complex presentation goal into presentation tasks. It also decides whether an object description is to be uttered verbally or graphically. The result is a presentation script that is passed to the the Visual Information and Natural Language generation modules.

The visual generation module creates the visual arrangement of the content using dynamically created and filled graphical layout elements. Since many objects can be shown at the same time on the display, the manager re-arranges the objects on the screen and removes objects, if necessary. The visual structure of the user interface (UI) is defined in an Android-based multimodal application by means of layouts. Layouts can be defined by declaring UI elements in XML or instantiating layouts elements at runtime. Both alternatives can be combined in order to declare the application's default layouts in XML and add code that would modify the state of the screen objects at run time. Declaring the UI allows to better separate the presentation of the application from the code that controls its behavior.

UI layouts can be quickly designed in the same way a web page is generated. Android provides a wide variety of controls that can be incorporated to the UI, such as buttons, text fields, checkboxes, radio buttons, toggle buttons, spinners, and pickers. The View class provides the means to capture the events from the specific control that the user interacts with. The user interactions with the UI are captured by means of event listeners. The default event behaviors for the different controls can also been extended using the class event handlers.

Natural language generation is the process of obtaining texts in natural language from a non-linguistic representation. The simplest approach consists in using predefined text messages (e.g., error messages and warnings). Finally, a text-to-speech synthesizer is used to generate the voice signal that will be transmitted to the user. We propose the use of the Google TTS API to include the TTS functionality in an application.

The text-to-speech functionality has been available on Android devices since Android 1.6 (API Level 4). To listen a sample of the included TTS speech synthesizer, once located in the settings menu of the device, the option Settings of Speech Synthesis must be selected in the menu Speech Input and Output. This menu allows selecting the TTS engine, language, and speed used to read a text (from very low to very fast).

The *android.speech.tts* package includes the classes and interfaces required to integrate text-to-speech synthesis in an Android application. They allow the initialization of the TTS engine, a callback to return speech data synthesized

by a TTS engine, and control the events related to completing and starting the synthesis of an utterance, among other functionalities.

3.1 Modeling User's Intention

The statistical technique that we propose to model user's intention is described in [11]. The proposed technique carries out the functions of the ASR and SLU modules, i.e., it estimates user's intention providing the semantic interpretation of the user utterance in the same format defined for the output of the SLU module. A data structure, that we call *User Register* (UR), contains the information provided by the user throughout the previous history of the dialog. For each time i, the proposed model estimates user's intention taking into account the sequence of dialog states that precede time i, the system answer at time i, and the objective of the dialog $\mathcal{O}$. The selection of the most probable user answer U_i is given by:

$$\hat{U}_i = \arg\max_{U_i \in \mathcal{U}} P(U_i | UR_{i-1}, A_i, \mathcal{O})$$

The information contained in UR_i is a summary of the information provided by the user up to time i. That is, the semantic interpretation of the user utterances during the dialog and the information that is contained in a user profile (e.g., user's name, gender, experience, skill level, most frequent objectives, additional information from previous interactions, user's neutral voice, and additional parameters that could be important for the specific domain of the system). We propose to solve the previous equation by means of a classification process, which takes the current state of the dialog (represented by means of the set $UR_{i-1}, A_i, \mathcal{O}$) as input and provides the probabilities of selecting the different user dialog acts.

3.2 Fusion of Input Modalities and Dialog Management

The methodology that we propose for the multimodal data fusion and dialog management processes considers the set of input information sources (spoken interaction, visual interaction, external context, and user intention modeling) by means of a machine-learning technique. As in our previous work on user modeling and dialog management [11], we propose the definition of a data structure similar to the *User Register* to store the values for the different concepts and attributes provided by means of the different input modalities along the dialog history, which we called Interaction Register (IR).

The information contained in the IR at each time i has been generated considering the values provided by the input modules of the system along the dialog history. Each slot in the IR can be usually completed by means of more tan one input modality. If just one value has been received for a specific dialog act, then it is stored at the corresponding slot in the IR using the described codification. Confidences scores provided by the modules processing each input modality are

used in case of conflict among the values provided by several modalities for the same slot. Thus, a single input is generated for the dialog manager to consider the next system response.

As in our previous work on dialog management [12], we propose the use of a classification process to determine the next system response given the single input that is provided by the interaction register after the fusion of the input modalities and also considering the previous system response. This way, the current state of the dialog is represented by the term (IR_i, A_{i-1}), where A_{i-1} represents the last system response. The values of the output of the classifier can be viewed as the a posteriori probability of selecting the different system responses given the current situation of the dialog, as the following equation shows:

$$\hat{A}_i = \arg\max_{A_i \in \mathcal{A}} P(A_i | IR_i, A_{i-1})$$

4 Practical Application: An Advanced Multimodal City Street Guide for Android-Based Mobile Devices

We have applied our proposed framework to develop a practical multimodal system acting as an enhanced city street guide service for Android-base mobile devices. The app can be operated either visually or orally and is able to locate interesting sites near the current position of the user or a different starting point indicated by the user. It is able to locate sites such as banks, libraries or restaurants and to retrieve and display information about these sites, visualize their position in different maps, show routes, visit their webpages or phone them. The system is also able to initiate navigation to a selected spot considering different means of transportation, and to track the position where the user has parked and show a route to it if needed. This information is provided in Spanish.

To offer these functionalities the system uses Google Maps, Google Directions and Google Places. Google Maps Android API makes it possible to show an interactive map in response to a certain query. It is possible to add markers or zoom to a particular area, also to include images such as icons, highlighted areas and routes. Google Directions is a service that computes routes to reach a certain spot walking, on public transport or bicycle, and it is possible to specify the origin and destination as well as certain intermediate spots. Google Places shows detailed information about sites corresponding to number of categories currently including 80 million commerces and other interesting sites. Each of them include information verified by the owners and moderated contributors. The application also employs the android.speech and android.speech.tts libraries described in the previous section.

When the application is started, it displays a map centered in the current location of the user. The user can search for a place in three ways: spotting it with a finger in the map, introducing the address in a text field, indicating it orally. In any of the three cases, the user is indicating a destination, which will be marked in the map by the system with a red sign. The system also shows the

route to the destination from the current location. It can be shown visually or orally, and it is possible to set the preferred transportation means.

The application offers the possibility to look for stores in a long list of options around the user position or a position selected previously. The search can be performed by touching the screen, using the graphical interface or orally. Once the stores are retrieved, e.g. restaurants in an area of 1km around the campus (Figure 2, left), the user can obtain further information about them. When a store is selected, the view is centered on it and an information box appears indicating the name of the store and its address (Figure 2, center). A new screen contains an HTML block comprised of an image representing the type of store, its name, geographic coordinates, complete address, punctuation in the Google+ social network, telephone, website, and its profile in Google+ (Figure 2, right).

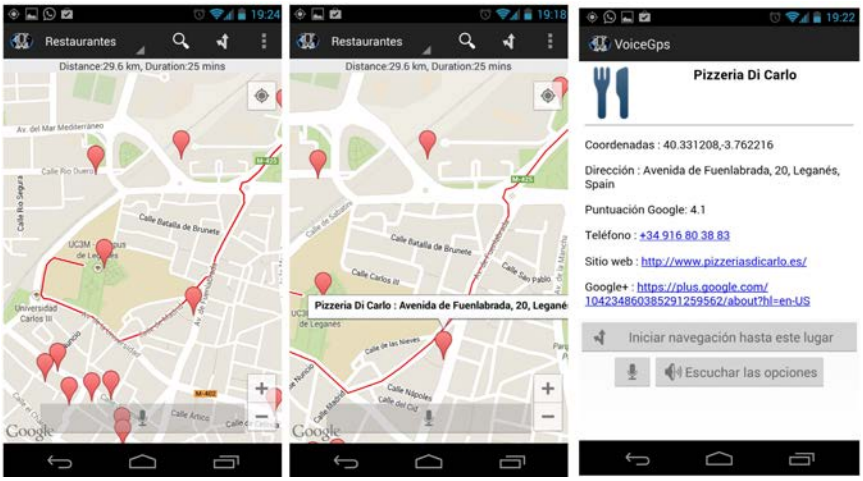

Fig. 2. System functionality to look for specific places and show the corresponding information

Finally, it is possible to store the location where the user parked his vehicle in order to be able to track it. Initially, the user must register the location using a drop-down menu with the visual option "I have parked here" or uttering this sentence after touching the microphone button. This way, the application stores the coordinates in which the user is at the moment and inserts a blue marker in the map. When the user wants to go back to the vehicle, he can press the option "Where is my car?" or utter the same sentence. Then, the application centers the map in the location registered indicating how to get there as shown in Figure 3.

The statistical models for the user's intention recognizer and dialog management modules were learned using a corpus acquired by means of an automatic

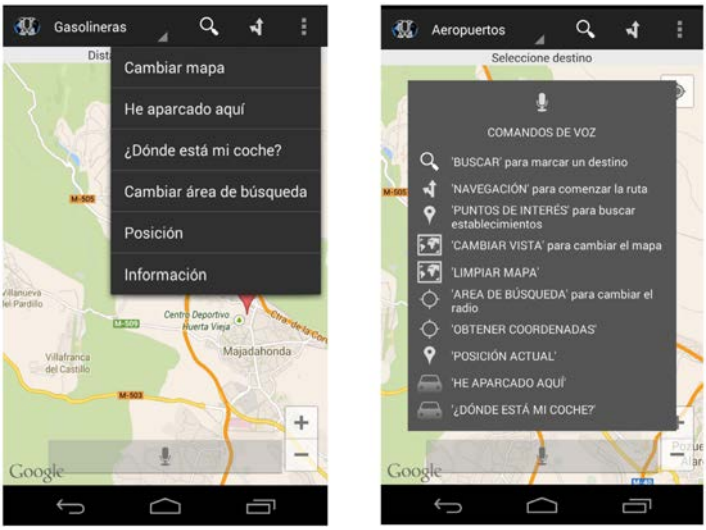

Fig. 3. "Where is my car?" functionality and configuration options for the system

dialog generation technique previously developed [13]. The application also allows users to complete a profile corresponding to their preferences on the location of the initial maps, preferred travel facilities, preferred types of stores, and specific details for each one of them.

5 Preliminary Evaluation and Discussion

We have already completed a preliminary evaluation of the developed system with recruited users and a set of scenarios covering the different functionalities of the system. A total of 150 dialogs for each agent was recorded from the interactions of 25 users. We asked the recruited users to complete a questionnaire to assess their opinion about the interaction. The questionnaire had seven questions: i) Q1: *How well did the system understand you?*; ii)Q2: *How well did you understand the system messages?*; iii) Q3: *Was it easy for you to get the requested information?*; iv) Q4: *Was the interaction with the system quick enough?*; v) Q5: *If there were system errors, was it easy for you to correct them?*; vi) Q6: *How did the system adapt to your preferences?*; vi) Q7: *In general, are you satisfied with the performance of the system?* The possible answers for each questions were the same: *Never/Not at all*, *Seldom/In some measure*, *Sometimes/Acceptably*, *Usually/Well*, and *Always/Very Well*. All the answers were assigned a numeric value between one and five (in the same order as they appear in the questionnaire).

Also, from the interactions of the users with the system we completed an objective evaluation of the application considering the following interaction parameters: i) question success rate (SR), percentage of successfully completed

questions: system asks - user answers - system provides appropriate feedback about the answer; ii) confirmation rate (CR), computed as the ratio between the number of explicit confirmations turns and the total of turns; iii) error correction rate (ECR), percentage of corrected errors.

Table 1 shows the average results of the subjective evaluation using the described questionnaire. It can be observed that the users perceived that the system understood them correctly. Moreover, they expressed a similar opinion regarding the easiness to understand the system responses. In addition, they assessed that it was easier to obtain the information specified for the different objectives, and that the interaction with the system was adequate and adapted to their preferences. An important point remarked by the users was that it was difficult to correct the errors and misunderstandings generated by the ASR and NLU processes in some scenarios. Finally, the satisfaction level also shows the correct operation of the system.

The results of the objective evaluation for the described interactions show that the developed system could interact correctly with the users in most cases, achieving a success rate of 96.73%. The fact that the possible answers to the user's responses are restricted made it possible to have a very high success in speech recognition. Additionally, the approaches for error correction by means of confirming or re-asking for data were successful in 94.15% of the times when the speech recognizer did not provide the correct input.

Table 1. Results of the preliminary evaluation with recruited users (For the mean value M: 1=worst, 5=best evaluation)

Q1	M = 4.56, SD = 0.47
Q2	M = 4.67, SD = 0.35
Q3	M = 4.12, SD = 0.58
Q4	M = 3.74, SD = 0.39
Q5	M = 3.49, SD = 0.51
Q6	M = 3.97, SD = 0.55
Q7	M = 4.02, SD = 0.27

SR	CR	ECR
96.73%	11.00%	94.15%

6 Conclusions and Future Work

Multimodal interactive systems offer the user combinations of input and output modalities for interacting with the systems, taking advantage of the naturalness of speech. In particular, multimodal interfaces are a useful alternative to graphic user interfaces for mobile devices, allowing the use of other communication as an alternative to tapping through different menus. However, there are no guidelines for the development of multimodal interfaces for mobile devices. Different vendors offer APIs for the development of applications that use speech as a possible input and output modality, but developers have to design ad-hoc solutions

to implement the interaction management. In this paper we have presented a general-purpose modular framework for the development of mobile speech applications in Android that integrates the libraries provided by the Android API.

Using our framework it is possible to develop multimodal interfaces that optimize interaction management and integrate different sources of information that make it possible for the application to adapt to the user and the context of the interaction. To show the pertinence of our proposal, we have implemented an evaluated an Android application that uses geographical context in order to provide different location services to its users. The results show that the users were satisfied with the interaction with the system, which achieved high performance rates. We are currently using the framework to build applications in other increasingly complex domains implying different web services and web services mashups.

References

1. Pieraccini, R.: The Voice in the Machine: Building Computers That Understand Speech. MIT Press (2012)
2. Agree, E.: The potential for technology to enhance independence for those aging with a disability. Disability and Health Journal 7(1), 33–39 (2014)
3. McTear, M., Callejas, Z.: Voice Application Development for Android. Packt Publishing (2013)
4. Vesnicer, B., Zibert, J., Dobrisek, S., Pavesic, N., Mihelic, F.: A voice-driven web browser for blind people. In: Proc. of Interspeech/ICSLP, pp. 1301–1304 (2003)
5. Tsai, M.: The VoiceXML dialog system for the e-commerce ordering service. In: Proc. of CSCWD 2009, pp. 95–100 (2005)
6. Kearns, M., Isbell, C., Singh, S., Litman, D., Howe, J.: CobotDS: A Spoken Dialogue System for Chat. In: Proc. of AAAI 2002, pp. 425–430 (2002)
7. Nishimoto, T., Kobayashi, Y., Niimi, Y.: Spoken Dialog System for Database Access on Internet. In: Proc. of AAAI 1997 (1997)
8. Polifroni, J., Chungand, G., Seneff, S.: Towards the Automatic Generation of Mixed-Initiative Dialogue Systems from Web Content. In: Proc. of Eurospeech 2003, pp. 193–196 (2003)
9. Gabbanini, F., Burzagli, L., Emiliani, P.: An innovative framework to support multimodal interaction with Smart Environments. Expert Systems with Applications 39, 2239–2246 (2012)
10. Corchado, E., Wozniak, M., Abraham, A., de Carvalho, A., Snásel, V.: Recent trends in intelligent data analysis. Neurocomputing 126, 1–2 (2014)
11. Griol, D., Carbó, J., Molina, J.: A statistical simulation technique to develop and evaluate conversational agents. AI Communication 26(4), 355–371 (2013)
12. Griol, D., Callejas, Z., López-Cózar, R., Riccardi, G.: A domain-independent statistical methodology for dialog management in spoken dialog systems. Computer Speech and Language 28(3), 743–768 (2014)
13. Griol, D., Carbó, J., Molina, J.: An Automatic Dialog Simulation Technique to Develop and Evaluate Interactive Conversational Agents. Applied Artificial Intelligence 27(9), 759–780 (2013)