Multimodal Framework for Mobile Interaction

Francesco Cutugno  Vincenza Anna Leano  Gianluca Mignini
LUSI-Lab@ University of Naples  LUSI-Lab@ University of Naples  LUSI-Lab@ University of Naples
“Federico II”  “Federico II”  “Federico II”
Via Cinthia I  Via Cinthia I  Via Cinthia I
80126 Naples, Italy  80126 Naples, Italy  80126 Naples, Italy
cutugno@unina.it  vincenzaanna.leano@unina.it  gianlucamignini@gmail.com

Roberto Rinaldi
LUSI-Lab@ University of Naples
“Federico II”
Via Cinthia I
80126 Naples, Italy
rober.rinaldi@studenti.unina.it

ABSTRACT

In recent years multimodal interaction is becoming of great interest thanks to the increasing availability of mobile devices. In this view, many applications making use of speech, gestures on the touch screen and other interaction modalities are presently becoming to appear on the different app-markets. Multimodality requires procedures to integrate different events to be interpreted as a single intention of the user. There is no agreement on how this integration must be realized as well as a shared approach, able to abstract a set of basic functions to be used in any possible multimodal application, is still missing. Designing and implementing multimodal systems is still a difficult task. In response to this situation, the goal of our research is to explore how a simple framework can be used to support the design of multimodal user interfaces.

In this paper we propose a framework that aims to help the design of simple multimodal commands in the mobile environment (more specifically in Android applications). The proposed system is based on the standard licensed by the W3C consortium for the Multimodal Interaction [8] [9] and on the definition of a set of CARE properties [2]. The pioneer of multimodal interface systems was Bolt with his “Put that there” system [1] in which graphical objects were created and moved on a screen using voice recognition and finger pointing. In 1995 a set of theoretical guidelines were defined that were named CARE Proprieties (Complementary, Assignment, Redundancy, Equivalence) [2]. These properties established which modes of interaction between users and systems are available, and, at the same time, helped to formalize relationships among different modalities. More recently Dumas et al. [3] introduced SMUIML (Synchronized Multimodal User Interaction Modeling Language): a language representing multimodal interactions.

Some works attempted to introduce multimodal interaction on web browser based systems. For example Honkala & Poljä [4] introduced XFormsMM, an attempt to extend XForms [5], in order to obtain both graphical and voice interfaces on a web browser. The basic idea was specifying the abstract controls with XForms elements and using CSS2 aural and visual for the vocal and graphical rendering. The problem is that the CSS aural has limited ability in terms of voice interaction and the proposed solution requires an ad hoc environment and a browser in order to work. In [6] authors presented XISL (eXtensible Interaction Scenario Language): a markup language based on XML for the definition of the web multi-modal interaction systems. XISL allows the description of multi-modal input/outputs synchronization and control of the dialogue flux and transition. The language permits the separation of the content (defined in the HTML file) from the interaction (defined in the XISL document). The problem introduced by this language is that it needs its own

Keywords: Multimodal Interfaces, Mobile Interaction Design, Map-based Interaction

1. INTRODUCTION

In this paper we propose a framework that aims to help the design of simple multimodal commands in the mobile environment (more specifically in Android applications). The proposed system is based on the standard licensed by the W3C consortium for the Multimodal Interaction [8] [9] and on the definition of a set of CARE properties [2].

Categories and Subject Descriptors

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AVI '12, May 21-25, 2012, Capri Island, Italy
Copyright © 2012 ACM 978-1-4503-1287-5/12/05... $10.00.
interpreter and its own execution environment. One of the first attempts of introducing multimodal interaction on mobile systems was the QuickSet [7] system, developed by the US Marine Corps: a multi-modal collaborative platform that allows the management and configuration of military simulations. With the widespread adoption of touch screens on mobile devices, mobile broad band and speech recognition, it is only now that interfaces supporting true multimodal commands are available to everyday users. An example, is the Speak4it local search application [18], where users can use multimodal commands combining speech and drawing to issue mobile search queries. An important contribution has been given by the W3C which has recently established a set of multi-modality guidelines. In [8], [9] the requirements identified by the W3C for multi-modal interaction and the architecture proposed by the consortium were defined. The guidelines consist of a series of proprieties and theoretical standards that multi-modal architecture should follow and implement.

The system proposed in this paper is based on the assumption that managing the different interactions of a multi-modal interface is not an easy task for HCI designers. Unlike GUI based systems that offer a one way interaction guided by forms and buttons, the presence of several way of interacting with a multimodal interface present difficulties for the relative formalization and the management.

In order to simplify the design of a multi-modal interface, an approach that conforms to the W3C Multi-modal Framework [8] was adopted which looks at the proprieties of the relationship between different modes of interaction as specified in [2]. We propose a framework which allows to design a complex multimodal interaction between the user and the system using an XML file. In order to model the multi-modal commands according to the CARE proprieties, an approach that uses NFA (Nondeterministic Finite Automaton) was chosen. The framework is application-independent and can be easily adapted to support application-specific tasks as well as new interaction modalities. It manages multimodal interaction in four steps:

1. obtaining data from input interaction modalities,
2. extracting semantics from the received data,
3. fusing data in order to carry out user intent,
4. sending a message to the application client business logic in order to perform an action based on the recognized user intention.

This framework has been designed to be plugged into applications needing to receive notifications from multimodal input events generated by a set of modality recognizers.

With reference to [8], [9], the presented framework manages the Fusion process (i.e. how to fuse data received by various modalities of interaction to carry out user intent), while the Fission process (i.e. how to split the output data between modalities) should be managed by the application client.

2. THE PROPOSED SYSTEM

2.1 Architecture

The architecture for the multi-modal interface management of the examined case study is showed in Figure 1, it is W3c Multimodal Framework [8] compliant. The architecture provides a communication between the various interaction modalities and the Interaction Manager (the core of the framework proposed) based on asynchronous events. Every mode of interaction is composed of two subsections:

- Input Recognizer: which focuses its attention on a single mode source of information like voice, gesture, rfid, touch pad etc... Therefore it captures the input from the user and transforms it in a machine-readable form.
- Semantic Interpretation Component: which manages and keeps constantly observed its own Input Recognizer in order to verify its progress. It performs a semantic extraction of data received and forwards the extracted data to the Interaction Manager.

The data sent by the Semantic Interpretation Component are received by the Interaction Manager that, thanks to an XML file (see Sec. 2.3) which describes how to merge data received by various modalities, combines the several Interpretation Components outputs. Then the Event Notifier Component encapsulates the merged data and sends them to the client application that use the framework, which will be notified of the incoming data by means of an event listener. Session Component manages the user session interaction and helps the Interaction Manager to perform the fusion of the semantic data received from Semantic Interpretation Components.

2.2 Interaction Manager Core

In this section a description of how the presented framework models the multimodal command by a NFA is provided.

In the automata theory, an NFA representing a regular language can be used in an accepting mode to verify if an input string is admitted in that language [13]. In the accepting mode an input string is provided and the automaton can read it from left to right, one symbol at a time. If all input symbols have been processed and the system is in an accept state, then we know that the input string was indeed part of the language, and it is said to be accepted, otherwise it is not part of the language and it is not accepted [13]. In our framework we use a similar technique in order to:

1. merge the various input modalities that the Interaction Manager receives,
2. accept a multimodal command;
3. describe how a particular set of modal inputs provided by the user can be used to activate a set of multimodal commands.
All the events generated by the user interface (speech, touch, etc...) are sent asynchronously by the corresponding Semantic Interpretation component to the Interaction Manager which is related to an active NFA. The data received by the Interaction Manager can trigger a transition to the active NFA that follows the action, if a matching transition was found. When the NFA arrives in an accepting state, the user-intended multimodal command is activated and the framework notifies the application client of the behavior to carry out with the needed data. The problem now is: how to create the NFA to model multimodal commands? Just as it was specified in [10], where authors used FSM (Finite State Machine) to implement a toolkit to evaluate the usability of multi-modal interfaces, FSMs in general are useful for modeling multimodal commands. As described in [10] for example, the command “moving an object” may be specified by the following sequence of inputs:

- mouse-press on an object, mouse-drag and mouse-release;
- mouse-press on an object, mouse-release, speech input “move” and mouse-press mouse-release on a target position.

Such sequences of inputs and actions can be modeled by simple finite state machines (FSMs) (See Figure 2).

FSMs are also very useful to specify the CARE properties of multimodal commands as reported in [14]. The problem is that we don’t know a priori which command the user wants to send to the application and how he/she wants to do that. So the framework doesn’t know which FSM should use to validate the events that will be received and if it should accept the command. To resolve this problem we transform the FSM described above into a NFA having its first state set to a nondeterministic status. With reference to the “moving an object” example in Figure 2 we don’t know if, to perform the action, the user will use the first sequence or the second so we model this with a NFA like in Figure 3.

When the action is activated by the NFA the framework notifies to the application client the behavior to carry out with the needed data to perform that action. In the example the application client will receive the action to do (move an object), the coordinate x-y of the destination target and the coordinate x-y of the object to move or the id of that object to move. This is developer-dependent because he/she is who decides the list of events that trigger transactions in the NFA through an XML configuration file of the framework. The NFA also is created automatically by the framework thought an XML configuration file.

2.3 Framework XML Configuration File

Let's look at the details of how the configuration of the framework is described with an XML file used by the Interaction Manager and how it models multimodal command using a NFA.

The XML configuration file allows the developer to:

1. specify which events received by the Interaction Manager could trigger a transition in the NFA;
2. specify which multimodal command the user can activate;
3. specify how he/she can activate it using various modalities.

This file uses some feature of the SMUIML (Synchronized Multimodal User Interaction Modeling Language) [3], [16]. SMUIML is an XML based design language oriented to the description of human-computer multimodal interaction. In this work we used only a small amount of its features expressly taking into account our needs to interface the interaction design and commands available to the user with the NFA modeling the integration and the semantics. In this view we outperformed the SMUIML potentialities adding to this language a back-end interface toward the implementation lay-out. As already said, we used this language to model the Multi-modal User Interface because it provides developers with a easy to read source code, it is very expressive to describe the multi-modal interaction and allows to model the user-machine dialogue and the CARE properties. We use the `<recognizers>` block section of the SMUIML to specify the various modality of interaction allowed to the user. For each modality a list of variables that can be attached, the `<triggers>` block, are used to define a list of events to be sent to the client application. These triggers are the input events giving rise to a state transition in the related NFA. The `<dialog>` block is then used to describe which multimodal command the user can activate and how he can do this. In the `<dialog>` tag any `<context>` element represents a NFA that the developer can activate any time.
during the execution of his application to recognize some specific multimodal commands. A given number of <command> elements can be attached to a given <context> tag, every <command> tag represents a multimodal command that the user can perform. The developer can use the <command> tag to describe how any command can be activated by the user (i.e. which triggers have to be generated by the user to activate that multimodal command) and pinpoint time synchronicity issues using <complementary>, <assignment> and <equivalence> elements inside the <command> tag that are the CARE proprieties (except redundancy). Inside the <command> tag the developer has to specific a mandatory <result> tag that describes the command name and which parameters will be notified to the application client when the multimodal command is activated by the user. An example of this XML file is given in Table 1.

Table 1. Example XML modeling the “moving object” multimodal command

```
<recognizers>
  <recognizer modality="speech">
    <variable name="operation"/>
  </recognizer>
  <recognizer modality="mouse">
    <variable name="x_target_pos" value="[1-200]"/>
    <variable name="y_target_pos" value="NOTNULL"/>
    <variable name="id_target_obj" value="[1-200]"/>
  </recognizer>
  <variables name="moving"/>
</recognizers>

<triggers>
  <trigger name="mouse_target_obj" modality="mouse">
    <variable name="id_target_obj" value="[1-200]"/>
  </trigger>
  <trigger name="mouse_target_pos" modality="mouse">
    <variable name="x_target_pos" value="NOTNULL"/>
    <variable name="y_target_pos" value="NOTNULL"/>
  </trigger>
  <trigger name="mouse_move" modality="mouse">
    <variable name="moving" value="true"/>
  </trigger>
  <trigger name="speech_move" modality="speech">
    <variable name="operation" value="move|shift"/>
  </trigger>
  <trigger name="speech_clean" modality="speech">
    <variable name="operation" value="clean|trash|delete"/>
  </trigger>
</triggers>
<dialog>
  <context id="1">
    <command leadtime="5000" name="move_object">
      <complementary>
        <trigger name="mouse_target_obj" />
        <trigger name="mouse_move" />
        <trigger name="speech_move" />
      </complementary>
      <trigger name="move_object">
        <variable name="id_target_obj", 
        x_target_pos,y_target_pos"/>
      </trigger>
    </command>
    <command name="clear">
      <assignment>
        <trigger name="speech_clean"/>
      </assignment>
    </command>
  </context>
</dialog>
```

As shown in Table 1 there is one context that can be enabled by the developer, in this context there are two commands available. Inside the first <command> tag the developer specifies complementary time synchronicity issues. In this example we want that the user, in order to activate the multimodal command “moving object”, firstly uses the mouse to identify the target moving object, successively he/she can use either the mouse or the speech (saying move or shift) to choose the moving operation. With the help of this platform we have a ready to use standard architecture for the design and implementation of multi-modal applications available. The platform will permit a quicker development of the interface and an easier maintenance.

3. MOBILE GIS PROTOTYPE

The case study here chosen to show the advantages of using a multi-modal approach to mobile user interface for GIS system queries, is the creation of an informative map-based system for XXX (city name omitted for blind review) public transportation (henceforth YYY, public bus company name omitted for blind review). The system provides the user the option of making multi-modal queries to obtain information regarding public transportation, such as:

- seeing a map that shows all of the bus stops of the chosen transportation line;
- seeing the arrival times of transportation lines at a chosen bus stop;
- seeing additional information relative to a bus stops such as a list of transportation lines that transit at the chosen stop and all of the temporary detours on the line;
- seeing a map with the nearest bus stops to the current location;
- seeing a map with the bus stops on a given street or area of interest

3.1 Mobile GIS Architecture

Getting results from spatial queries implies the elaboration of a great quantity of data. Taking into consideration the physical limitations and the computational power of modern mobile devices, it's not possible to implement these operations on the device itself, and therefore the chosen architecture for the service is of the client-server type, so that the calculations (spatial queries) can be made on the server, limiting the mobile device to sending requests and processing the replies so that they can be displayed on the map. In order to deal with the huge amount of geographic data on the server, a Spatial Database is absolutely necessary in order to index the data and allow spatial queries (for example coverage or intersection between two geometries). The geometries available in the spatial database are Multi-Line to represent the roads in the area of interest and Points to represent the bus stops. As far as the functions implemented by the server, only one spatial operator is provided: coverage. Specifically the operator handles the shape drawn by the user and returns the objects within it.

A SQLite database [11] is available in the mobile application and contains all the information relative to the spatial position of the bus stops, the active public transportation lines, possible detours, user preferences etc....
The information regarding arrival predictions for the following 30 minutes at a given bus stop is retrieved from the public transportation company website by making an http-post request through a web-service offered by YYY and parsing the html response received.

3.2 Case of Study

In this section we describe how the prototype of the case of study proposed was implemented on Android platform [15] on a mobile device (tablet or smartphone) equipped with a touch screen. A Google Maps [12] integrated service was used in the Android environment and the touch screen of the terminal was used to specify spatial queries.

Moreover, for this example of use, we used the Input Recognizer and the Semantic Interpretation component for speech offered by Loquendo Mobile Embedded ASR Platform. This tool is embedded in the Android OS and makes use of ABNF grammars and a proprietary tag set for semantics (Loq-semantics). These tag sets are usually linked to the grammars in order to relate uttered sentences to actions required.

The application allows the following operations:

- **Search by line**: allows the user to see all the bus stops of a public transportation line on the map.
- **Search by bus stop number**: allows the user to know the arrival time predictions of the various bus lines by specifying the number of the bus stop of interest.
- **Favorite stops**: allows the user to see predicted arrival times for his favorite bus stops which were previously specified.
- **Browse stops**: allows a user to visualize a list of the routes and bus stops of the various public transportation lines.
- **Search by street name**: allows the user to visualize on a map the bus stops on a specified street.
- **Settings**: allows the user to configure some of the application's settings like the default transportation line to be displayed on the map.
- **Vocal commands**: by clicking on the microphone icon present on the right of the title bar, it's possible to interact with the application through vocal commands.

Let us go into the details of the functionality of 'Search by line', which represents in some ways the heart of the application. As shown in Figure 5, the user sees a local map relative to his current position with markers which represent the stops made by various lines of the service.

Once the stops of interest are displayed on the map, the user can get various information for a bus stop simply by tapping on the corresponding marker and a balloon can be recalled containing expected arrival times. By default a balloon will be displayed with predicted arrival times (Figure 6), or the user can see a list of lines that make stops at the bus stop of interest or a list of detours currently in effect. By using the star icon (Figure 6) it's possible to insert a stop into a favorite stops list so that the user can quickly access the information he is interested in. Every stop added to the favorites list has an alias defined by a user in order to simplify its identification.

3.2.1 Multimodal commands in the application

The mobile application allows the fruition of some content by vocal and multimodal command. A vocal interaction is started by a click on the microphone icon.

There are three voice-activated functions:

1. shows a transportation line on the map;
2. loads the arrival time predictions for a chosen stop;
3. knows the wait time of a chosen line at a chosen stop.
User can perform also a multi-modal query on the map drawing a location on the map and uttering the action to be done in that location by voice (see Figure 8).

The implemented drawing functions support point and poly-lines. To be more specific: a point is when the user touches a specific object (e.g. the bus stop) on the mobile map; a poly-line is when the user swipes the screen over the map. The implementation of these query interfaces is relatively straightforward. We attach two layers to the touch screen: one at the bottom is the map layer and one at the top is an empty layer, which is used to capture any touch gestures from the user. The top layer is the most important as it converts the touch points into a latitude/longitude coordinates set transmitted to the underlying base map. Audio is processed by the speech platform which first performs speech recognition and then extracts some semantic knowledge to understand which action to do, the gesture is recognized by the specific recognizer and performs a geo-coding of the touch points on the map (see Figure 9). These data then is processed by the Multimodal Framework (cfr. 2).

The system sends the latitude/longitude on the line drawn on the maps to the GIS server and then returns the bus stops closest to the route drawn. The Figure 7 shows the NFA that models some multimodal commands available in the prototype system.

4. CONCLUSIONS

In this paper we have presented a framework that aims to help the designing and implementation of a multimodal interface with relatively little effort. The system is presently receiving an accurate testing in order to verify its usability and its stability in different mobile connections conditions. Future works will include the use of machine learning techniques to improve the process of user’s intent estimation [17].

5. REFERENCES


international conference on Web engineering, pages 201–208, New York, NY, USA, 2006. ACM.


[18] Patrick Ehlen, Michael Johnston, Multimodal Local Search in Speak4it, Proceedings of the 16th international conference on Intelligent user interfaces (2011)