Architecture of Multi-Modal Dialogue System

Martin Fuchs, Petr Hejda, and Pavel Slavík

Department of Computer Science and Engineering,
FEE, Czech Technical University,
Karlovo nám. 13,
12 135 Praha 2, Czech Republic
slavik@cslab.fel.cvut.cz
phejda@ra.rockwell.com
http://www.cgg.cvut.cz

Abstract. This article describes an architectural framework of a multi-modal dialog system. The framework is based on separation between the semantic and the syntactic parts of the dialog. The semantics of the human-computer conversation is captured using a formal language. It describes the conversation by means of sequences of dialog elements. The paper further elaborates how to derive the syntactic features of the conversation. A two-layer architecture has been proposed for the dialog system. The upper layer, called sequencer, works with the description of the whole dialog. The lower layer (driver dock) deals with individual dialog elements.
A prototype has been implemented to demonstrate the main benefits of our framework. These are adaptability and extensibility. The adaptability involves multiple modes of communication, where the modalities can be changed even during the course of a dialog. Due to layering, the applications can be easily extended to additional modes and user interface devices.

1 Introduction

This article describes an architecture framework of a multi-modal dialog system. In general, a multi-modal dialog system takes advantage of the multi-sensory nature of humans. It can utilize more than one sense for communication. Traditionally, applications are based on visual modes. Additional modes can be, for example, speech, gesture, and tactile input/output modes. The multi-modal approach increases the bandwidth of human-computer interaction and it may improve the interactive nature of the system. Also, it allows for deployment of the system in various environments.

Multi-modal systems may find their applications in a number of fields. To name only a few examples, they proved to be a viable aid for visually impaired users, an alternative to WIMP[1] interfaces in mobile computing, or an entertaining extension of computer games.

[1] Windows, Icon, Menu, and Pointer based interfaces

The functionality of a multi-modal system can be fairly extensive. We have focused on a simple, yet extensible dialog based system. The main purpose of the system is to present information to the user and to query him for data. The challenge was to build a system with the following features:

- **Simple Architecture** The architecture should be as simple as possible. It has to specify function blocks, their roles and type of data the blocks exchange.
- **Extensible** The system should be open to new communication devices. Once there is a need to connect a new device or use a new mode, the system should **not** be re-implemented.
- **Adaptive** The system should allow for changing of modalities. These changes should be possible even during the run-time of the system.

The text of this article is structured as follows. Section 2 introduces formal description of a human-computer conversation utilized in our approach. This formal description is used in the architecture as described in Section 3. Section 4 gives details of our system prototype. The last section summarizes our ideas and deals with the future development.

2 Formal Description of a Dialog

The framework focuses on multi-modal dialog systems in which the dialog consists of a finite number of dialog element types. In our previous work, we have shown that a class of dialogs can be described via sets of predefined, parameterized dialog elements [3]. These elements capture the semantics of the human-computer conversation. The elements can be divided into classes according to the type of information which is expected as the result of the conversation with the user.

An example of an element can be a simple question, such as *"Would you like to continue?"*. It can be captured by a dialog element with following formal description[2]:

```
<question id="2" default="yes">
    Would you like to continue?
</question>
```

The description captures three pieces of information:

1 The **request** is a question sentence it "What ..."
2 The **answer** *"yes"* or *"no"* is expected
3 The **default value** is *"yes"*

These three pieces capture the semantics of the dialog element.

The dialog system has to add syntax of the actual conversation with the user. This may vary according to the device and mode used for the communication

[2] This example uses XML, see bellow.

with the user. For example, the **request** can be displayed together with a check box. The check box is initially checked to reflect the **default value**. In another situation, the speech recognition and synthesis is used. In that case, the system voices the request to the user and expects one of the words it "yes", *"no"*, *"default"*, *"whatever"*, *"yep"*, *"o.k."* etc ... As a result, the dialog system can have some freedom in interpreting the dialog elements and handling the actual conversation with the user.

As mentioned above, the dialog system receives the formalized dialog description. In our prototype, Dialog Markup Language version 2.0 (DML2.0) was used [1]. It can describe the following types of dialog elements:

1 **Message.** It presents some information to the user.
2 **Question.** A yes/no answer is expected.
3 **Choice.** An array of choices. The user can chose one or several of them.
4 **Text.** Single-line text. Allows for editing linear text.
5 **TextEdit.** Multi-line text.
6 **Number.** A number is expected as a result.
7 **Program.** Any sequence of simple dialog elements.
8 **Dialog.** Any sequence of simple dialog elements and programs.

Each simple dialog element (elements 1–6) represents a part of a conversation focused on delivering or obtaining a basic piece of information, such as a number or a text string. Their semantics is standardized. The last two items in the list describe sequences of simple dialog elements. Their semantics is defined as a sequence of joint semantics of used elements. Hence, the structure and the semantics of the whole dialog are described in a formal way [2].

3 Architecture of the System

The formal description of the dialog can be used for communication between the user interface part and the application part of the system. This creates a natural division of the application in two sub-systems. The **application side** generates a dialog to obtain some data from the user, for example what to do next, name of a command or its parameters. The **user side** is then responsible for effective interpretation of the formal description. This section focuses on the architecture of the user side (see Figure 1).

The formalized description of the dialog is processed by the user part to work out the details of the conversation. During this process, it can be decided which modalities to use for which parts of the dialog.

The key concept of our architecture is the definition of higher and basic processing layers: the **sequencer** and the **low-level dock**. The sequencer transforms the DML dialog description into the description of single dialog elements. It processes the whole dialog at once, breaking it into dialog elements. This solves the counteracting nature of linear and non-linear conversation. The linear conversation requires processing of one dialog element at one time. An example can be voice-based conversation. The non-linear conversation, on the other hand,

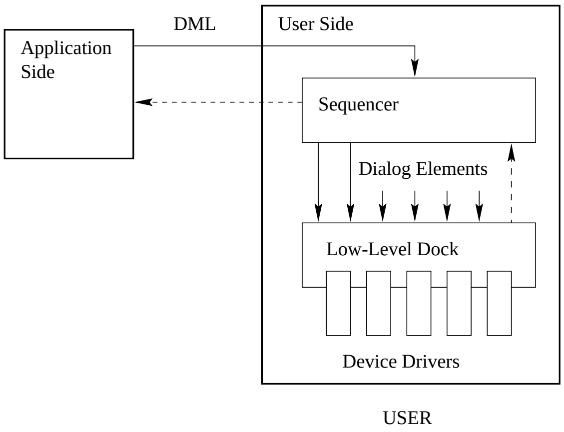

Fig. 1. The architecture of the framwork

processes dialog elements in parallel, as in case of a formular. The user is free to process the elements in any order [4]. The sequencer is the part of the dialog system responsible for making the conversation either linear or non-linear.

Figure 2 depicts a simple dialog which is processed in two ways. In the left-hand side, there is a non-linear conversation. The user is free to choose where to start. The right-hand side shows a sequence of frames featuring the same conversation in a linear fashion. Due to limitations of the small display, the dialog is broken into small pieces. These pieces are processed one by one, in a linear fashion. The figure was rendered by our prototype [1].

Even though the whole dialog system can be based on the sequencer layer, additional benefits are gained by employing the second layer, low-level dock. It defines a plug-and-play interface for software components processing the dialog elements on various devices using various modalities. The set of plug-ins defines an interface between device independent and device dependent parts of the user part.

For example, there are two modules allowing for the output to either an LCD display (as in Figure 2) or to a speaker (not depicted). The LCD module breaks any output message into small pieces which can fit the size of the display. To be able to use a speaker as an output, a voice synthesis module has to be used. Its main task is to transform all messages to speech and to repeat the missed parts.

4 The Prototype

To prove our ideas workable, we have implemented a working prototype of the application framework. Its structure reflects the architectural model described above. The framework is divided in two parts, the application and user sides. For testing purposes, the application side implements two applications.

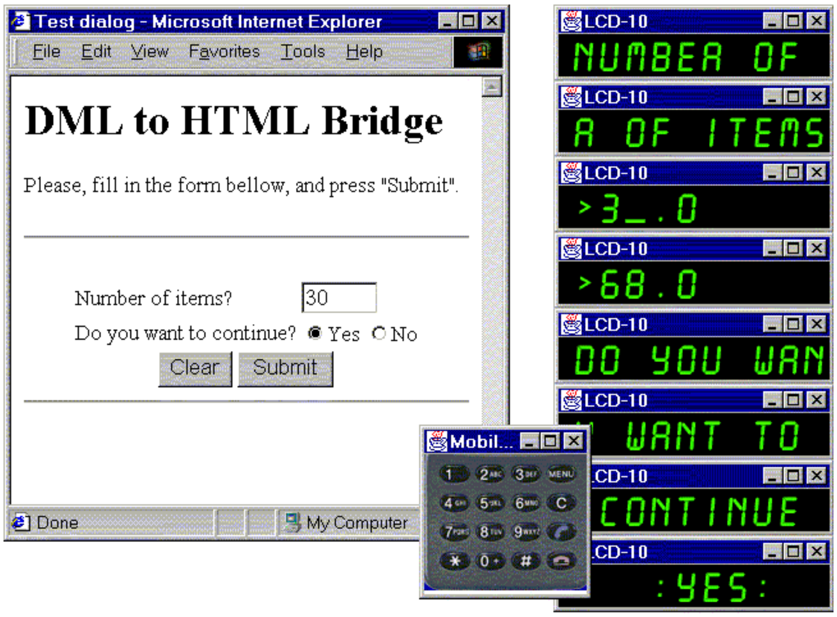

Fig. 2. The effect of the sequencer on the dialog structure

The first application is a reader of e-mail. It generates dialogs for reading and sending e-mail messages. The second is an automatic teller machine (ATM) simulator. The dialogs generated by this application mirror the functionality of a real ATM.

The user side was implemented according to the two-layer architecture. Where the sequencer provides the dialog sequencing, the low-level dock and its drivers process the dialogs and allow for various modes. The capabilities of this part are crucial to the functionality of the application.

The following drivers have been implemented for both applications:

– Term - text-based mode only. The processing is linear.
– HTML - text with more formatting. The processing is non-linear.
– Applet - widget based interface. The processing is non-linear.
– Keyboard - driver for text input.
– MobileKeys - driver simulating phone keyboard.
– Mouse - driver for input using a mouse.
– X-Key - driver for a joystick-like device with two buttons.
– SmallLCD - driver for a small alphanumeric display.
– SmallBitmap - driver for a small bit mapped display.
– Voice - driver for speech synthesis and recognition.

The drivers allow for mutual combination and switching during the execution of the application. This results in a highly flexible user interface, which is capable of changing modes of interaction. Although the drivers cover only a small portion of possible input/output devices, it is apparent the drivers are fairly small and easy to implement components.

The implementation of the prototype was done using the Java 1.1[3] and Extensible Markup Language (XML) technologies[4]. The Java language brings additional benefits of portability and platform independence. Both the application and the user side were implemented as separate components connected via a data stream.

DML was implemented as an application of XML. This allowed for fast prototyping using the SAX compliant XML parser Aelfred. Speech synthesis and recognition was implemented with IBM's ViaVoice[5] and SpeechForJava API[6].

5 Summary and Future Work

We have described an architectural framework of a multi-modal dialog system. It is based on the formal dialog description in DML. The formal description captures the semantics of the conversation using dialog elements. The syntactic features are delivered by our two-layer architecture of the dialog system. The upper layer, called sequencer, works with the description of the whole dialog. The lower layer deals with individual dialog elements. As a result, we have got an application framework which allows for changes of modalities even during the processing of a dialog. The other benefit is that the applications can be easily extended to additional modes and user interface devices.

In the future, we would like to extend the framework to cover a wider set of applications and drivers. The other issue is usability testing and evaluation of the framework.

References

1. Fuchs, M.: Multi-modal user interface (in Czech). Master thesis, Department of Computer Science and Engineering, FEE, Czech Technical University Prague (1999)
2. Hejda, P.: Dialog Markup Language. Poster99, FEE, Czech Technical University in Prague (1999) IC8
3. Hejda, P.: Situations in Dialogs. In Text, Speech and Dialogue. Springer-Verlag, Berlin Heidelberg New York (1999) 290–295
4. Larson, J. A.: Interactive Software. Yourdon Press (1992)

[3] See `http://java.sun.com/`.
[4] See `http://www.w3.org/XML/`.
[5] See `http://www-4.ibm.com/software/speech/`.
[6] See `http://www.alphaworks.ibm.com/tech/speech`.