Human-Computer Interaction -- INTERACT'03
M. Rauterberg et al. (Eds.)
Published by IOS Press, (c) IFIP, 2003, pp. 717-720

Designing and Prototyping Multimodal Commands

Marie-Luce Bourguet

Queen Mary, University of London, Mile End Road, London, E1 4NS, UK

mlb@dcs.qmul.ac.uk

Abstract: Designing and implementing multimodal applications that take advantage of several recognition-based interaction techniques (e.g. speech and gesture recognition) is a difficult task. The goal of our research is to explore how simple modelling techniques and tools can be used to support the designers and developers of multimodal systems. In this paper, we discuss the use of finite state machines (FSMs) for the design and prototyping of multimodal commands. In particular, we show that FSMs can help designers in reasoning about synchronization patterns problems. Finally, we describe an implementation of our FSM-based approach, in a toolkit whose aim is to facilitate the iterative process of designing, prototyping and testing multimodality.

Keywords: multimodal commands, synchronisation patterns, finite state machines, design, prototyping, toolkit

1 Introduction

Multimodal interaction refers to interaction with the virtual and physical environment through natural modes of communication such as speech, body gestures, handwriting, graphics or gaze. Recent developments in recognition-based interaction technologies (e.g. speech and gesture recognition) have opened a myriad of new possibilities for the design and implementation of multimodal systems. However, our lack of understanding of how these new modes of interaction can be best combined in the user interface often leads to interface designs with poor usability.

In order to help designers, some attempts have been made to elicit relationships between different interaction techniques. The CARE properties for example (Coutaz, 1995) are a framework for reasoning about multimodal interaction from both the user and the system perspectives. But, however useful this framework may be, it does not offer rapid and practical solutions to designers. Moreover, multimodal systems must be equipped with adequate software architectures to combine the different modalities. Unfortunately, current models of architecture, such as (Nigay, 1995) and (Oviatt, 2000) are too generic and complex to serve as prototyping tools. Designing and implementing multimodal systems is still a difficult task. In response to this situation, the goal of our research is to explore how simple modelling techniques and tools can be used to support the designers and developers of multimodal user interfaces.

As a starting point, we have implemented a toolkit whose aim is to facilitate the design and prototyping of simple multimodal commands. According to our definition, a multimodal command is a combination of several user inputs, used to activate a particular function of an application (e.g. a function for moving graphical objects on a computer display). The user inputs that enter in the expression of a multimodal command may belong to different modalities (e.g. speech and gesture). In this paper, we show that Finite State Machines (FSMs) constitute a good framework for describing multimodal commands and for combining sets of user inputs of different modalities. In particular, we show that FSMs can help designers in reasoning about synchronisation patterns problems.

2 Modelling Multimodal Commands Using Finite State Machines

Finite State Machines are a well-known technique for describing and controlling dialogs in graphical user interfaces (Wasserman, 1985). A FSM typically consists of states, events, transitions and actions (see Figure 1). A transition has a source and a target state and is executed when the FSM is in the source state

and the event associated with the transition occurs. Upon the execution of a transition, an action associated with it can be triggered. Typically, a machine is initially in a "start" state, as user inputs arrive, they are compared against the transitions leaving the current state. If the event matches the transition, the FSM moves to the state at the other end of the transition (target state); if no matching transition is found, the FSM usually moves to a special error state.

Figure 1: Finite State Machine (FSM)

According to (Hudson, 1992), controlling a complete dialog with FSMs can present some significant drawbacks (such as promoting the use of modes), but FSMs are very appropriate for controlling dialogs at the command level. We show here that FSMs are also useful for modelling multimodal commands.

Figure 2 illustrates how different multimodal commands can be modelled by simple FSMs. The first FSM (Figure 2) represents a multimodal command for "moving an object" that is specified by the following sequence of inputs: mouse-press on an object, a mouse-move (eventually followed by several optional mouse-move) and a mouse-release. Alternatively, the second FSM represents another multimodal command to activate the same function, but characterized by a different sequence of inputs: mouse-press then mouse-release on an object, speech input "move", and finally mouse-press then mouse-release on a target position.

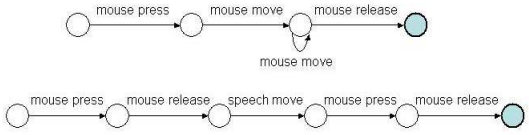

Figure 2: Two different multimodal commands to describe the function "moving an object".

When designing multimodal commands, one important task is the specification of the synchronization requirements. The aim is to guarantee that users will be able to activate the commands in a natural and spontaneous manner. In human-human interaction, the temporal synchrony of speech and hand gestures has been analysed for different languages. In human-computer interaction however, little experimental evidence is currently available for reasoning about synchronization requirements (Oviatt, 1997), (Bourguet, 1997). In practice, a user can produce inputs in a sequential (e.g. with pen input completed before speech begins) or simultaneous manner (when both inputs show some temporal overlap).

FSMs constitute a good framework for testing different synchronization patterns. For example, Figure 3 describes a speech and pen "move" command where many different synchronisation patterns are represented. The top branch of the FSM (Figure 3) starts with a "speech move" event, allowing users to initiate the command using speech. In contrast, the bottom branch starts with a "mouse-press" event, allowing users to use the pen first. This initial "mouse-press" event can then be followed by either a "mouse-move" input or a "speech move" input. According to this representation, users are thus given the possibility to organize their sequence of inputs in several ways and following different synchronization patterns. In other words, with such a representation of a command, users are free to deliver inputs in their preferred order (sequentially or simultaneously, pen first or speech first).

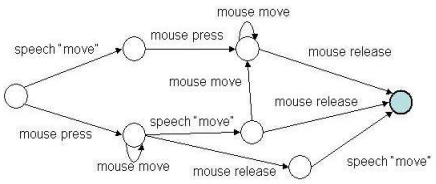

Figure 3: FSM modelling a "move" speech and pen command where many synchronisation patterns are represented. The interaction is unconstrained.

Figure 4 shows another FSM where only one synchronisation pattern is now allowed. According to this FSM, users must use the pen first and pronounce the "speech move" event before the sequence of pen inputs is finished. Such an FSM has for effect to constrain users in their usage of the modalities.

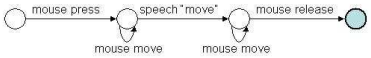

Figure 4: FSM modelling a "move" speech and pen command where only one synchronisation pattern is allowed. The interaction is constrained.

We have used FSMs to test the designs of several graphical applications. For example, for the purpose of studying spontaneous synchronization patterns for different interaction styles, we have implemented an application in which the users were asked to place famous London landmarks on a map. Four different interaction styles were implemented: (1) "speech-gesture" commands (e.g. to make a picture appear on the map, the user draws a "P" and pronounces the name of the landmark); (2) "speech-point" commands (e.g. to resize a picture, the user points at a picture and says "smaller"); (3) "speech-point-point" commands (e.g. to move an image, the user says "move it here", points at an image and points at the destination) and (4) "speech-drag" commands (e.g. to move an image, the user says "move" and drags the image to its desired location). Different synchronization patterns were incrementally tested to determine the best possible synchronization requirements, according to each interaction style.

3 Toolkit

We have integrated our FSM-based framework in a toolkit that aims at facilitating the iterative process of designing, implementing and testing multimodal commands (Bourguet, 2002). Low-level APIs with the recognition engines and user inputs management are both handled in the toolkit in a transparent manner. Iterative design and declaration of FSMs is done via a simple graphical editor.

Figure 5 shows the global architecture of the system. The toolkit comprises two main modules: the "Interaction Model" and the "Multimodal Engine" to which is connected the core of the application (where the functions of the application are implemented).

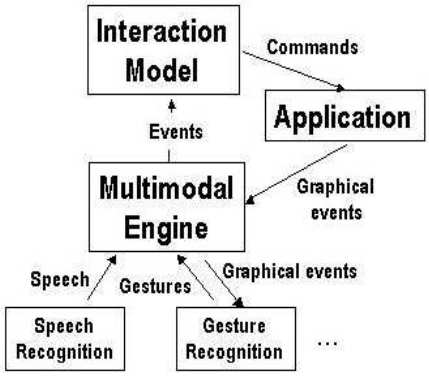

Figure 5: Toolkit global architecture.

The Interaction Model contains all the FSMs to be tested. Multimodal commands can be tested one by one (i.e. the interaction model contains only one FSM), or several of them can be tested simultaneously. There is no constraint on the number of FSMs that an interaction model contains.

The Multimodal Engine is responsible for controlling the different input channels and recognition systems (e.g. speech and gesture recognition systems) and for managing user inputs. Managing user inputs includes (1) formatting them so they can be matched with transition events in the interaction model, (2) handling and dispatching recognition hypotheses as they are delivered by the recognition engines. As input management is the entire responsibility of the Multimodal Engine, the application (e.g. a graphical application) is not allowed to handle itself any user input. In consequence, it is required to send all graphical events (e.g. mouse press and menu selection) to the Multimodal Engine.

The Multimodal Engine dispatches user inputs to all the FSMs of the interaction model. In order to match a user input with the event of a transition, the individual elements that compose the event of the transition are parsed against the constituents of the user input. Typically, an input starts with the modality type (e.g. speech, mouse, pen, gesture, etc.). A speech input may then contain a tag (e.g. colour), followed by the word or sequence of words that was pronounced (e.g. "green"). Similarly, a gesture input may contain a gesture class (e.g. command) and a gesture name (e.g. delete). A mouse input contains an action type (press, release or move), followed by the x and y coordinates of the pointer. When an FSM enters a state with which an action is associated, it communicates to the application the complete set of user inputs that allowed it to move from its start state to its current state. The application is then in charge of evaluating the action and of deciding whether it should be executed or not. For example, when the user presses a mouse button (transition event), an action "select object" together with the complete user input (including the coordinates of the pointer) is sent to the application. The application and only the application is then able to decide whether an object should be selected or not.

In complement with the toolkit, we have developed a graphical editor, called IMBuilder, which supports the task of designing and declaring interaction models. With IMBuilder, new interaction models can be created or existing ones can be

uploaded for modification (iterative design). The declaration of an FSM is entirely done graphically. New states and transitions can simply be drawn in the editor, while events and actions are typed or selected from a list provided with the corresponding application. Each interaction model is saved in an XML file for subsequent use and testing with the Multimodal Engine. The Multimodal Engine is implemented in Java. XML is used to describe interaction models, which are then reconstructed and implemented in Java as described in (VanGurp, 1999).

4 Conclusion

The iterative design, implementation and testing of multimodal user interfaces is difficult due to a lack of supporting tools for designers and developers. In response to this, we have developed a toolkit, based on a simple formalism (FSMs) that aims at facilitating this process. We discussed in this paper how FSMs can be used to describe and test multimodal commands, act as modality integrators, and help reasoning about synchronization problems.

Currently, the task of designing and declaring the interaction models remains the responsibility of the designers. One way of speeding up the design process and insuring that interaction models are realistic is to automatically generate them from experimental observations. This constitutes our future work. Potential users will be required to freely produce inputs with the aim of activating specific application functions. These inputs will then form the basis for the automatic generation of FSMs.

Acknowledgements

This research is supported by the Nuffield Foundation under grant NUF-NAL 00.

References

Bourguet, M.L. (2002), A Toolkit for Creating and Testing Multimodal Interface Designs, *Poster in companion proceedings of UIST 02* (Paris, October 2002), 29-30.

Bourguet, M.L. and Ando, A. (1997), Speech timing prediction in multimodal human-computer interaction, *in proceedings of INTERACT'97* (Sydney, July 1997), Chapman & Hall, 453-460.

Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J. and Young, R. (1995), Four easy pieces for assessing the usability of multimodal interaction: the CARE properties, *in Proceedings of INTERACT'95* (Lillehammer, June 1995)

Hudson, S. and Newell, G. (1992), Probabilistic State Machines: Dialog Management for Inputs with Uncertainty, *in Proceedings of UIST'92* (November 1992), 199-208.

Nigay, L et al (1995), A Generic Platform for Addressing the Multimodal Challenge, *in Proceedings of CHI 95* (Denver, May 1995), ACM Press, 98-105.

Oviatt, S. et al. (1997), Integration and synchronization of input modes during multimodal human-computer interaction, *in Proceedings of CHI '97* (Atlanta, March 1997), ACM Press, 415-422.

Oviatt, S. et al (2000), Designing the User Interface for Multimodal Speech and Gesture Applications: State-of-the-Art Systems and Future Research Directions, *in Human Computer Interaction*, Volume 15, No 4, 263-322.

Van Gurp, J. and Bosch, J. (1999), On the implementation of finite state machines. *In Proceedings of the IASTED International Conference* (Scottsdale, Arizona, October 6-8, 1999).

Wasserman, A. (1985), Extending State Transition Diagrams for the Specification of Human-Computer Interaction, *IEEE Transactions on Software Engineering*, Volume 11, No 8, (August 1985), 699-713