A Toolkit for Creating and Testing Multimodal Interface Designs

Marie-Luce Bourguet
Queen Mary, University of London
Mile End Road, E1 4NS, London, UK
+44-20-7882-5200
mlb@dcs.qmul.ac.uk

ABSTRACT
Designing and implementing applications that can handle multiple recognition-based interaction technologies such as speech and gesture inputs is a difficult task. IMBuilder and MEngine are the two components of a new toolkit for rapidly creating and testing multimodal interface designs. First, an interaction model is specified in the form of a collection of finite state machines, using a simple graphical tool (IMBuilder). Then, this interaction model can be tested in a multimodal framework (MEngine) that automatically performs input recognition (speech and gesture) and modality integration. Developers can build complete multimodal applications without concerning themselves with the recognition engine internals and modality integration. Furthermore, several interaction models can be rapidly tested in order to achieve the best use and combination of input modalities with minimal implementation effort.

KEYWORDS: Multimodal inputs, interaction model, finite state machines, prototyping

INTRODUCTION AND MOTIVATION
With the deployment of modern interaction technologies such as speech and gesture recognition on every desktop machine, most user interfaces will become multimodal in the near future. The availability of multiple interaction techniques opens a myriad of new possibilities for the design of user interfaces. However, our lack of understanding of how these interaction techniques can be best combined often leads to interface designs with poor usability. Designing and implementing multimodal user interaction is still a difficult task. Some attempts have been made to elicit relationships between different interaction techniques and propose design criteria for multimodal applications [1], but iterative design and implementation remain critical to the success of a multimodal user interface.

Currently available commercial prototyping tools (such as Visual Basic™) do not support the integration of modern, recognition-based, input technologies. The goal of this research is to explore how new tools can be used to support the design and prototyping of multimodal user interfaces. We have developed a toolkit that allows a developer to (1) specify different multimodal interaction designs using a simple graphical tool (IMBuilder) and (2) use and test these designs in a multimodal framework (MEngine) that automatically performs input recognition and modality integration.

INTERACTION MODELS
We define an interaction model as a description of how a particular set of modal user inputs can be used to activate a set of multimodal application commands. For example, in one interaction model, the command "moving an object" may be specified by the following sequence of inputs: mouse-press on an object, mouse-drag and mouse-release. Alternatively, in another interaction model, the same command could be specified by a different sequence of inputs such as: mouse-click on an object, speech input "move" and mouse-click on a target position.

Such sequences of inputs and actions can essentially be modelled by simple finite state machines. Finite state machines provide a powerful way to describe dynamic behaviour of systems. A state machine consists of states, events, transitions and actions. A transition has a source and a target state and is performed when the state machine is in the source state and the event (i.e. user input) associated with the transition occurs. Upon the execution of a transition, an action (i.e. command) associated with the transition can be executed. Figure 1 shows how the command "moving an object" can be modelled by a finite state machine.

THE INTERACTION MODEL BUILDER
IMBuilder is a graphical tool that allows the specification of multimodal interaction models (see Figure 1). Through the "Application" menu, a list of valid user inputs and application commands can be created or imported from an XML file provided by the application. From the "Interaction Model" menu, a new model can be created or an existing model can be imported from an XML file, opened, modified and saved again.

When the user selects the option "New" from the "Action Unit" menu, a partially built finite state machine appears on the screen. Each machine comprises at least four states: an empty state (the machine is waiting for all other system modules to initialise); a start state (the machine is waiting for a user input); a sleeping state (the machine is waiting for a "reset" event in order to return to the start state) and an active state (the machine has successfully been through all

transitions and is waiting for a "reset" event in order to return to the start state; this state is represented by a red circle). In the simplest case, the user only has to add one transition between the start state and the active state. Adding a transition can easily be done by pressing the mouse on the source state, and then dragging and releasing the mouse onto the target state. A dialog window opens where the event (i.e. a user input in any interaction modality) and the action associated with the transition can either be typed in or selected from a list. When a new state must be added to the machine, the green square in the top left corner of the screen can be dragged anywhere on the screen and becomes a new state (white circle) when released. An unlimited number of states can be added to a machine, depending on the complexity of the command to be modelled. Figure 1 shows a finite state machine for the command "moving an object", which is made of 7 states and 8 transitions.

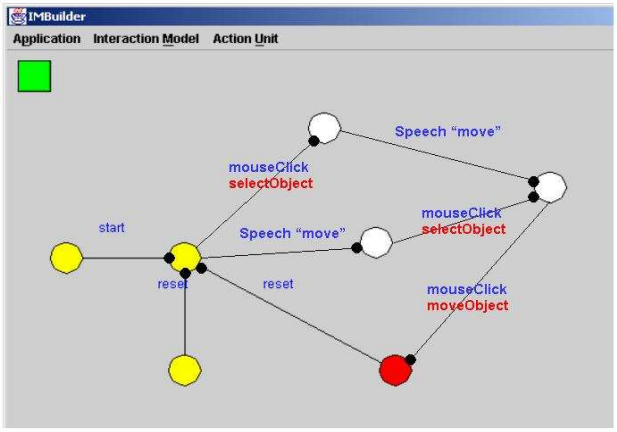

Figure 1: A finite state machine to model the command "moving an object", as generated with IMBuilder.

The finite state machines are implemented as described in [3]. Each interaction model is saved in an XML file for subsequent use and testing with MEngine.

TESTING AND USING AN INTERACTION MODEL

Interaction models can then be tested using MEngine, a multimodal framework whose architecture is shown in Figure 2. MEngine currently implements the IBM ViaVoice™ speech recognition engine and the Satin toolkit [2] for 2D gesture recognition. Any other recognition technologies could be used instead.

The application communicates to the Multimodal Engine the address of different resource files: the XML file that describes the Interaction Model to be tested, the speech grammar file to be used by the speech engine (if relevant) and the gesture training models to be used by the gesture recogniser (if relevant). The Multimodal Engine is responsible for starting the recognition engines and for re-constructing the Interaction Model (collection of finite state machines) from its XML description.

As shown in Figure 2, user actions on the application graphical user interface (graphical events such as mouse press and menu selection) are sent to the Multimodal Engine. Every event (graphical, spoken or gestural) is then dispatched by the Multimodal Engine to the Interaction Model. The Interaction Model holds a reference to the current state of each finite state machine. When applicable, transitions are executed and commands are sent to the application, in charge of their evaluation and execution.

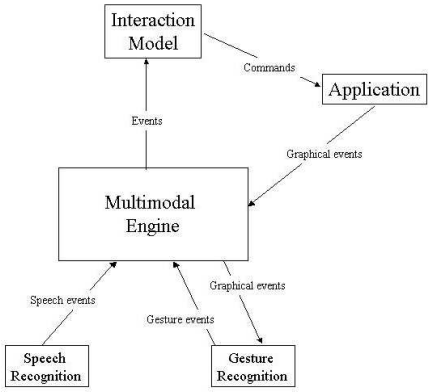

Figure 2: The MEngine architecture.

EXPERIENCE USING THE TOOLKIT

We have used the toolkit to design and test multimodal drawing applications. In particular, different interaction techniques (direct manipulation, speech and gestures) could be quickly implemented and their respective advantages and disadvantages evaluated for the specification of commands such as moving, resizing and deleting objects. These applications will be used to illustrate our poster. Another advantage of the toolkit that was discovered while experimenting with it is the possibility of sharing interaction models across applications to achieve better consistency of interaction styles between applications.

Future work on the toolkit includes improving the efficiency of the event-dispatching mechanism and clearer interfacing between the application and MEngine. In its improved form, we believe that the toolkit presented here could then be used not only for prototyping and evaluation purposes but also for the final implementation of multimodal interfaces.

ACKNOWLEDGEMENT

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

REFERENCES

1. Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J. and Young, R. Four easy pieces for assessing the usability of multimodal interaction: the CARE properties. In Proceedings of INTERACT'95 (June 25-29, 1995, Lillehammer, Norway)
2. Hong, J. and Landay, J. Satin: a toolkit for informal ink-based applications. In Proceedings of UIST'00 (November 5-8, 2000, San Diego, California) 63-72.
3. Van Gurp, J. and Bosch, J. On the implementation of finite state machines. In Proceedings of the IASTED International Conference (October 6-8, 1999, Scottsdale, Arizona).