Model-based Design and Generation of a Gesture-based User Interface Navigation Control

Sebastian Feuerstack, Mauro dos Santos Anjo, Ednaldo B. Pizzolato
Universidade Federal de São Carlos, Rodovia Washington Luís, km 235
São Carlos - São Paulo – Brasil
{sfeu, mauro_anjo, ednaldo}@dc.ufscar.br

ABSTRACT
The gesture-based control of interfaces could enable interaction in situations where hardware controls are missing and support impaired people where other controls fail. The rich spectrum of combining hand postures with movements offers great interaction possibilities but requires extensive user testing to figure out an optimal control with a sufficient control performance and a low error rate.

In this paper we describe a declarative, model-based gesture navigation design based on state charts that can be used for the rapid generation of different prototypes to accelerate user testing and comparison of different interaction controls. We use the declarative modeling to design and generate several variants of a gesture-based interface navigation control. The models are described using state charts and are transformed to state machines at system runtime. They can be directly executed to form a multimodal interaction.

Keywords
Model-driven design of user interfaces (MDDUI), Interface navigation, Hand-Gesture Recognition, HCI.

1. INTRODUCTION

Using gestures to control user interfaces could enable interaction in situations where hardware controls are missing, for instance, wall-sized displays [7] and could support disabled people to interact with a computer where other controls fail.

Different to common hardware supported interaction controls like mouse and keyboard setups or joysticks for instance, gesture interaction does not suffer from a predefined and limited command setup. The amount of possible gestures is only limited by the users' creativity. Gesture recognition has been practiced for a long time driven by e.g. software that detects coloured gloves or even the bare hands using video cameras.

Since we are interested in comparing different gesture-based interface navigation controls, we are confronted with the problem of quickly implementing different variants of interactions with the same application. Gesture-based interaction offers a rich set of possibilities by combining hand movements and postures to control an interface.

Therefore, we propose a new level of abstraction for constructing multimodal interfaces: declarative models to design and specify the way of interaction. Different to writing source code, declarative modelling is less technical and requires no programming skills. Further on, it eases the re-design of interaction to address e.g. different preferences or certain disabilities of users. We understand the declarative modelling as the next step towards end-user development that enables users without programming skills to change and configure their preferred ways of interaction.

In this paper we focus on presenting our approach of modelling gesture-based interaction based on state-charts that can be directly executed to run a multimodal interface. We used this approach to design and test several ways of navigating through an interface using gestures and postures. We present the three most promising designs to explain our modelling approach.

The paper is structured as follows: The next section discusses related work regarding frameworks to quickly prototype multimodal interaction and the model-driven design of user interfaces (MDDUI) in general. Section 3 presents our approach for designing gesture-driven interactions based on state-charts and gives details about the gesture recognitions (3.1), the modeling of the corresponding graphical user interface (3.2), as well as our notation for multimodal mappings (3.3) that can flexibly combine both to form a multimodal user interface. Finally, section 4 states future work.

2. RELATED WORK

Recent research has focused on evaluating different forms of gesture-based interaction to control interfaces displayed on wall sized displays [7]. Several frameworks have been proposed to ease the creation of multimodal interaction controls like the Open Interface Framework [9] or Squidy [8] for instance by assembling multimodal controls including gestures out of components. These frameworks focus on bridging different interaction technologies by

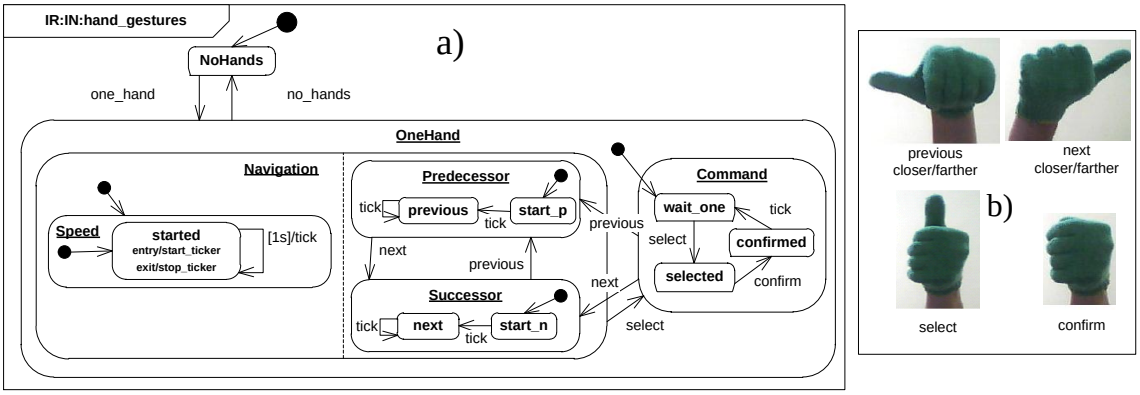

Fig.1. (a) The first variant of the gesture-based navigation control.
(b) The four basic gestures we selected to navigate through the interface.

enabling the interconnection of device drivers and signal processing algorithms to form new kinds of multimodal interaction setups. But the specification of how an application is controlled with an assembled interaction setup is still done at the source code level. Further on, the actual interaction inside the application that specifies what happens if the user issues a certain gesture in a certain state of the application is still a programming task as well.

The model-driven development of user interfaces has been around for a long time to tackle this issue and resulted in several connected design models that have been summarized by the CAMELEON Reference Framework [3] and by user interface languages such as USIXML [11]. But it has been applied to develop interfaces for pre-defined platforms only, such as to design interfaces for small screens of cell phones, for speech interfaces or to develop television and 3D interfaces for instance. Multimodal systems have been addressed by these approaches only to a very limited extend [13, 10].

These MDDUI approaches suffer from the fact that they introduce new languages and design processes through several abstract models that need to be learned by the designer. Additionally they require anticipation skills to understand how manipulations in the abstract model design are reflected in the final generated interface. Therefore, we decided to use state charts for interaction modelling, which has the advantage that they are already widely known and have a small sized basic vocabulary (mainly states and transitions driven by events).

State machines have been widely used in Case-Tools and are already standardized as part of UML and the W3C multimodal framework with the SCXML[1] standard and therefore reduce the entry barrier for developers as various tools are already available to design state machines.

[1] http://www.w3.org/TR/2011/WD-scxml-20110426/

3. MULTIMODAL INTERACTION DESIGN

Hand gestures are already widely used as a natural way of human-computer interaction [1, 2]. But the definition of suitable gestures depends on various factors and extensive user testing. These factors include for instance: the hand poses chosen, if one or both hands should be considered, the feedback of the interface when a gesture is recognized, delay on processing and communication, ergonomics, intuitiveness of the interaction, among other possible factors.

In our approach the interaction resources are declaratively modeled, directly executed and can be flexibly added and removed to the system by using the multimodal mappings. Therefore we argue that this approach eases to adapt the interface to different styles of gesture-based interaction considering, for instance, the actual situation of the user (is only one hand or are both hands available for interaction?), their age and training level (how fast should the gestures be interpreted?) or individual preferences about certain gestures (e.g. less exhausting or less explicitly gestures).

In our test cases we designed different gesture-based interactions and have considered four gestures that can trigger interface actions in two ways:

- A fixed hand posture: when the system recognizes a static gesture it triggers a single action and will wait for the next different posture to trigger the subsequent action. A fixed posture can have a temporal component like a ticker for instance to trigger the same event in fixed intervals.
- A motion-related gesture: When a certain gesture is recognized, the action triggered varies according to the movement detected.

3.1 Specification of a Gesture-and Posture Interaction Resource

We based the implementation of the gesture recognition on the project of finger spelling recognition of sign language [12]. The system is able to recognize gestures using colored gloves by doing HSV color space based segmentation. We used 25x25 pixels sized images of the gloves and trained an

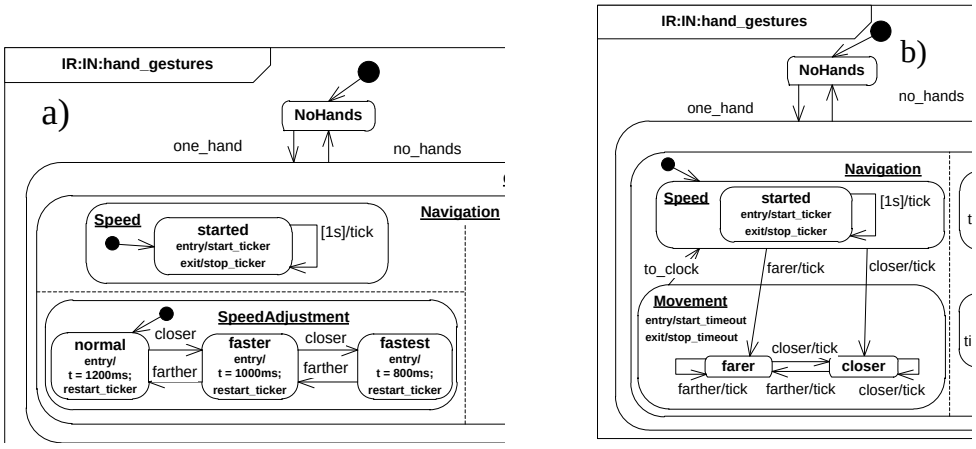

Fig.2. (a) Second Variant: To manipulate the navigation speed the user can move his hand closer to or farther away from the camera.
(b) Third Variant: Instead of a timed navigating step every second, by every hand movement closer or farther away from the cam a navigation step is done.

artificial neural network Multi-Layer Perceptron (MLP) with an architecture of 625x100x4 neurons in each layer. This network is able to classify the four gestures.

Figure 1a depicts a state chart that specifies our first variant of a gesture-based navigation control. It supports a basic movement to the next or the previous user interface element with the two gestures on top of figure 1b). The navigation speed is defined by a ticker that throws a "tick" event. For the first variant, we use a static ticker that throws a tick every second. Thus, if the user shows a "previous" gesture for instance, the state machine enters the super state "Predecessor" and starts with the initial "start_p" state. With every tick and as long as the user remains showing the previous gesture the "previous" state is entered with every tick event again and navigation step is performed by the interface. By showing the "select" gesture (figure 1b) the state machine switches to the "Command" mode, stops the navigation, and selects the actual user interface element as soon as the user confirms the selection with a "confirm" gesture.

The two state machines of figure 2 represent two navigation alternatives. For the sake of brevity, only the differences to the one of figure 1 are illustrated. Thus, the "Predecessor", "Successor", and "Command" states remain the same as already depicted in figure 1. The state chart of figure 2a) introduces an adjustable ticker. The time "t" between each "tick" could be adjusted by moving the hand that shows the previous or next gesture closer to or farther from the camera. Moving the hand closer to the camera results in a smaller ticker value (from 1200ms up to a speed of 800ms between the ticks). The variant 3 (figure 2b) additionally enables the user to switch between the timed ticker and by explicitly issuing ticks by moving the hand closer to or farther away from the camera. These movements temporarily disable the ticker (since it is not modeled as part of a parallel state) and enable a quicker navigation just by moving the hand.

To get the interaction running we need at least one mode to give the user a way to control the application and a media that presents the interface to the user. In this subsection we described the mode design, whereas in the following subsection we focus on a graphical, web-based presentation as an example for the media design.

3.2 Specification of graphical interface elements

Figure 3 shows a screenshot of the user interface that we used for testing the interface navigation. It shows an excerpt of a table of 5x5 cells, each consisting of a unique

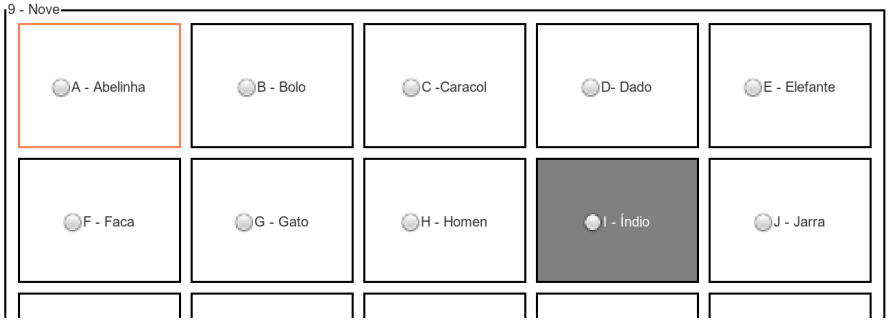

Fig.3. Detail of a screenshot from the user interface for testing the gesture-based navigation.

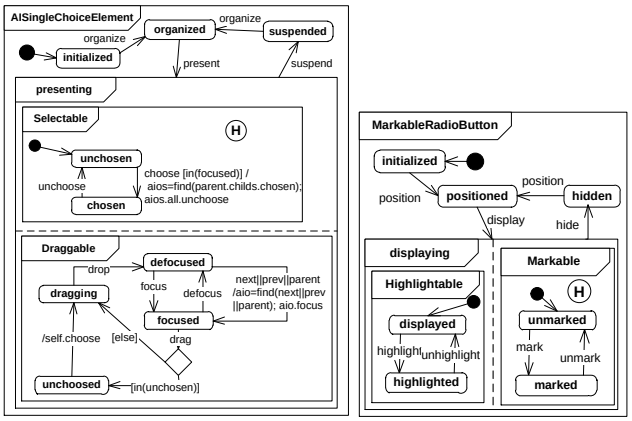

Fig.4. State chart of an abstract SingleChoiceElement (a) and an enhanced RadioButton that is "markable" (b).

letter and a radio box that could be selected. It was the users' task to navigate from the top-left box "A" to a cell that is marked with a grey background and select the button of this cell.

To specify this kind of interaction we use state chart models in the same way as we describe the posture recognition mode in the last section. But for an interface representation we use two models. One abstract model describing the interaction in a media independent way and a concrete model that considers specific features of a certain media. In our prototype of figure 4 the basic interaction element is a list that supports selecting one element at a time as well as navigating between its elements. Figure 4a shows the abstract specification of a list element whereas figure 4b adds its concrete specification for a graphical interface, describing a list element as a radio button that can be additionally "marked" (the grey background in figure 3 to set the navigation target for the user).

At runtime both state charts that complement each other, get instantiated as state machines (for each list element). The abstract part contains the core of the interaction (for the list element: to be able to be in the user's focus, to be selectable, and "draggable". The concrete part adds semantics relevant for graphical interfaces, such as it defines an element's focus (of the user) as "highlighting" (element A is highlighted in figure 3) or "positions" elements on an interface instead of just "organizing" them (organizing involves identifying an element's neighbours, whereas positioning refers to a graphical coordinate system). Finally, the specification of figure 4b adds the persistent (see history flag) "markable" feature to the radio button, which is specified by a parallel running "Markable" super state.

Both mode and media are glued together by multimodal mappings. These mappings are stored in a separate model, which enables changing them without touching the other models. Similar to the CARE properties they describe relations between different modes or media and are used to define basic multimodal fusion or fission.

3.3 Mappings

To define how the graphical interface of figure 3 should react upon gestures, we use multimodal mappings that observe state changes (stated by boxes with rounded corners) and generate events (defined by boxes with sharp edges) targeted to the graphical interface or to other media such as to generate sound. Figure 5 depicts two exemplary mappings that implement a basic fusion between the graphical interface and a certain gesture control, as well as a basic fission, which distributes the result to both the graphical presentation as well as plays a sound.

The first one (a) implements the navigation to the next element of the interfaces and plays a "tick" sound on each successful next movement. There are several operators that can be used to define mappings that we have presented earlier [4]. In this exemplary mappings the complementary operator, C, states observations that have to happen in a certain temporary windows (Tw) and the redundancy operator, R, publishes information to different media (such as changing to the next element and playing a "click" sound at the same time. The second mapping (figure 5b) depicts a mapping that implements a selection of an element of a list that is in the actual focus of the user.

Besides multimodal mappings we introduce synchronization mappings that are in charge of mediating between the abstract and the concrete media representation. Technically they communicate state changes between start machines and enable to synchronize different concrete medias (such as different graphical user interface formats) that share the same abstract model. Thus, in our example, a synchronization mapping connects bi-directionally the abstract "focused" state of figure 4a to the concrete "highlighted" state of the MarkableRadiobutton state chart (figure 4b). Every time the user issues a next posture and therefore triggers the mapping of figure 5a, the abstract "focus" is moved to the next element and is synchronized to "highlight" the corresponding RadioButton (figure 4b).

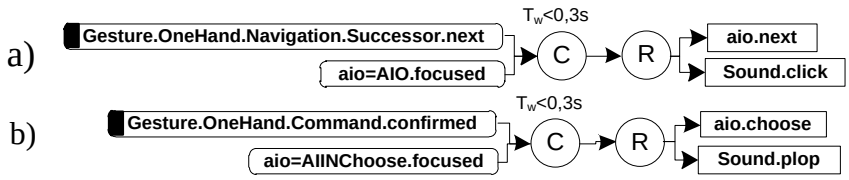

Fig.5. Mappings to connect sound media and a gesture-driven control to the interface.

4. CONCLUSIONS AND FUTURE WORK

With our approach by specifying interaction based on state machines, different forms of interactions could be efficiently designed and generated since only the state machines have to be changed. The mappings to describe the overall multimodal interaction have to be changed only if more than one mode of interaction needs to be manipulated. This was not the case in our prototypes, since we were focusing on comparing different gesture-based navigation alternatives without changing the other modes (the graphical interface or the sound feedback). We already applied the presented model-based design approach in several projects [4, 5] as well as performed user tests to compare the three different navigation variants that we described in the paper [6].

Differently to the state chart-based modelling of user interface elements that is already supported by tools, we rely on the SCXML standard and we are using a proprietary notation to design the mappings. Currently we are trying to formalize the notation and investigate in a suitable tool support that enables a developer to design and generate these mappings. Additionally, we are enhancing the notation to enable the design of more complex multimodal interactions where fusion data is retrieved by several different modes.

The software prototype as well as the entire abstract model specification can be downloaded from our website (http://www.multi-access.de).

ACKNOWLEDGMENTS

Sebastian Feuerstack is grateful to the Deutsche Forschungsgemeinschaft (DFG) for the financial support of his work.

REFERENCES

1. Joao Luiz Bernardes Jr., Ricardo Nakamura, and Romero Tori. Design and implementation of a flexible hand gesture command interface for games based on computer vision. In Proceedings of the 2009 VIII Brazilian Symposium on Games and Digital Entertainment, SBGAMES '09, pages 64–73, USA, 2009.
2. Lars Bretzner, Ivan Laptev, and Tony Lindeberg. Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering. In Proc. Face and Gesture, pages 423–428, 2002.
3. Gaelle Calvary, Joelle Coutaz, David Thevenin, Quentin Limbourg, Laurent Bouillon, and Jean Vanderdonckt. A unifying reference framework for multi-target user interfaces. Interacting with Computers, 15(3):289–308, 2003.
4. Sebastian Feuerstack, Ednaldo Pizzolato; Building Multimodal Interfaces out of Executable, Model-based Interactors and Mappings; HCI International 2011; 14th International Conference on Human-Computer Interaction; J.A. Jacko (Ed.): Human-Computer Interaction, Part I, HCII 2011, LNCS 6761, pp. 221--228. Springer, Heidelberg (2011), 9-14 July 2011, USA.
5. Sebastian Feuerstack, Allan Oliveira, Regina Araujo; Model-based Design of Interactions that can bridge Realities - The Augmented Drag-and-Drop;13th Symposium on Virtual and Augmented Reality (SVR 2011), ISSN 2177-676, 23th-26th May 2011, Brazil
6. Sebastian Feuerstack, Mauro Dos Santos Anjo, Jessica Colnago und Ednaldo Pizzolato; Modeling of User Interfaces with State-Charts to Accelerate Test and Evaluation of different Gesture-based Multimodal Interactions. Workshop:"Modellbasierte Entwicklung von Benutzungsschnittstellen (MoBe2011)", Informatik 2011, 4-7. October 2011, Germany
7. Fredrik Fikkert. Gesture Interaction at a Distance. PhD thesis, Universiteit Twente, Centre for Telematics and Information Technology, 2010.
8. Werner A. Kaenig, Roman Raedle, and HaraldReiterer. Interactive design of multimodal user interfaces - reducing technical and visual complexity. Journal on Multimodal User Interfaces, 3(3):197–213, Feb 2010.
9. Jean-Yves Lionel Lawson, Ahmad-Amr Al-Akkad, Jean Vanderdonckt, and Benoit Macq. An open source workbench for prototyping multimodal interactions based on off-the-shelf heterogeneous components. In EICS '09: Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems, pages 245–254, New York, NY, USA, 2009. ACM.
10. Grzegorz Lehmann, Marco Blumendorf, Sebastian Feuerstack, and Sahin Albayrak. 2008. Utilizing dynamic executable models for user interface development. In *T.C. Nicholas Graham and Philippe Palanque, editors, Interactive Systems – Design, Specification, and Verifications*. Springer Verlag Gmbh.
11. Quentin Limbourg, Jean Vanderdonckt, Benjamin Michotte, Laurent Bouillon, and Victor Lopez-Jaquero. USIXML: A language supporting multi-path development of user interfaces. In RemiBastide, Philippe A. Palanque, and Joerg Roth, editors, EHCI/DS-VIS, volume 3425 of Lecture Notes in Computer Science, pages 200–220. Springer, 2004.
12. Ednaldo B. Pizzolato, Mauro dos Santos Anjo, and Guilherme C. Pedroso. Automatic recognition of finger spelling for libras based on a two-layer architecture. In Proceedings of the 2010 ACM Symposium on Applied Computing, pages pages 969–973, 2010.
13. Adrian Stanciulescu, Quentin Limbourg, Jean Vanderdonckt, Benjamin Michotte, and Francisco Montero. A transformational approach for multimodal web user interfaces based on usixml. In ICMI '05: Proceedings of the 7th International Conference on Multimodal Interfaces, pages 259–266, New York, NY, USA, 2005. ACM Press