MINT-Composer – A Toolchain for the Model-based Specification of Post-WIMP Interactors

Sebastian Feuerstack
Universidade Federal de São Carlos
Rodovia Washington Luís, km 235
São Carlos - São Paulo – Brasil
sfeu@dc.ufscar.br

ABSTRACT

With the introduction of new modes to control user interfaces like speech, gesture and multi-touch Post-WIMP interfaces increasingly substitute classical WIMP (windows, icons, menus, pointer) user interfaces.

In this paper, we describe a toolchain of three connected tools to design and monitor custom Post-WIMP interactors using model-based user interface design (MBUID) methods. We use state charts and mappings as the basic modeling technique. Different to other MBUID approaches our models are get created to specify the behavior of existing widgets. These models are then used to attach further control modes and connect interactors using mappings.

We explain the toolchain by an interactive music sheet web application that can be controlled by head movements to turn the sheets.

Categories and Subject Descriptors

H.5.2 **[Information Interfaces and Presentation]**: User Interfaces - *Input devices and strategies, Interaction styles, Prototyping*; D.2.2 **[Software Engineering]**: Design Tools and Techniques – *User Interfaces.*

General Terms

Design, Human Factors, Languages.

Keywords

Model-based User Interfaces, Multimodal Interfaces, Web Application Development, HCI.

1. INTRODUCTION

Nowadays more and more devices get equipped with several sensors used to better capture the interaction with the user by considering the actual context of use. Whereas early cell phones and televisions used a numeric keypad to enable to navigate through the interface, modern smart devices offer a more natural form of interaction, involving a spoken control to search for or navigate through data or even gestures to control a game (e.g. the Kinect controller) or a television (using image processing by a camera).

The change to a more natural control has an impact on the graphical representation (the media) of the interface. The last 30 years a mouse and a keyboard were the principle devices to control an interface and there was consent that the best media to reflect these controls is a Windows, Icons, Menu, Pointer (WIMP) interface. With new devices, such as smart TVs and smart phones, tablets, wall-sized displays and multi-touch tables, the graphical media, which is still the most prominent output channel used – needs to be reconsidered to reflect these new and more natural forms of control.

A huge amount of Post-WIMP toolkits have been proposed to overcome the limitation of classic WIMP style interaction that does not consider tangible, gestural or speech-based interaction. But as of now there is no consent about the content of such a toolkit. Instead Post-WIMP toolkits are proposed for a specific form of interaction (e.g. for a multi-touch table, or for a specific smart phone type). Thus, cross-platform or cross-modality development of interactive systems to enable different user controls is still a difficult task.

In this paper we present a toolchain that enables a model-based specification of Post-WIMP interactors based on state charts and mappings. With our set of tools the behavior of pre-existing Post-WIMP interactors can be described to enable their re-use and extension to support new interaction modes.

The next section gives an overview about the development-process that is supported by the toolchain and presents the overall architecture. Thereafter, in section 3 we present the main functionality of the tools. Finally, in section 4 we discuss a use case where we applied the tool chain to design and implemented an music sheet Post-WIMP interactor that can be controlled by head movements. Section 5 states related work and section 6 concludes our work.

2. ARCHITECTURE

The process creating or modifying Post-WIMP interactors using our toolchain involves six basic steps that are depicted in figure 1:

1. Selecting a pre-designed interactor widget that should be used for the graphical presentation.
2. Specifying the behavior of the widget using a state chart.
3. Select and design the modalities that should be used for controlling the interactor by using a state chart.

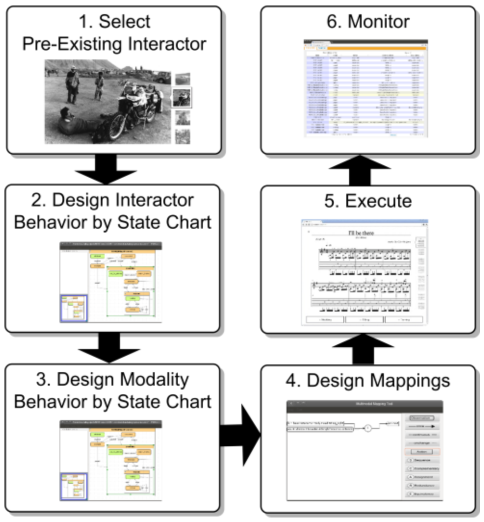

1. Select Pre-Existing Interactor
2. Design Interactor Behavior by State Chart
3. Design Modality Behavior by State Chart
4. Design Mappings
5. Execute
6. Monitor

Figure 1. **The interactor relations on the abstract level.**

4. Specification of the mappings that connect the widget with the modalities.
5. Executing of the interactors, modality and mappings specification within a runtime environment.
6. Realtime monitoring and debugging of the interactors and mappings during they are executed.

Our approach focuses on web interfaces, since there are already a broad range of widgets toolkits available that could be re-used. Therefore developers usually just need to select a widget from a toolkit that is already close to the requirements of the application.

After a widget has been chosen, its behavior relevant for the interaction with the user is described using a state chart editor. In our toolchain we use the scxmlgui editor that is available as open source and produces standardized W3C state chart xml (SCXML) files. Even if we focus on web widgets, the actual implementation language of an interactor is not important, but it is assumed that the widget offers an event-based API so that each state of the resulting start chart can be attached to a certain API call.

Different to the second step in that a media widget has been specified, the control of this media by the user depends on one or several modalities that are combined to form an interaction. Therefore in the third step state charts representing how a set of modalities can be used isolated (uni-modal) or combined (multi-modal) to control the Post-WIMP interactor are specified.

In the fourth step, multimodal mappings are defined that relate the graphical representation of an interactor with the control modes. The mappings are designed using a custom flow-chart like graphical notation based on the CARE properties [5] that describe multimodal relations like complementary, assignment, redundant, and equivalent relations. Technically mappings observe state changes and trigger actions by sending events to state machines. We implemented a tool, that we describe in the next section, to ease the specification of mappings.

In the fifth step all interactors are loaded in our runtime processing platform, the Multimodal Interaction (MINT) framework[1]. The platform compiles the Post-WIMP interactors and modality interactor state charts to state machines and the multimodal mapping to code based on the observer and publish/subscribe pattern and execute all components inside a web server.

Finally, in the last step a running Post-WIMP application can be observed by a monitoring application that is part of the MINT platform. The monitor observes and visualizes all running interactors and their state changes in real-time together with a mapping execution log to ease the understanding of which mappings are currently executed. The monitor helps to identify bugs or to improve the interaction by re-wiring the mappings or state charts of the interactors. Very complex state machines, such as used to describe gesture-based interactions, can even be observed directly inside the state chart editor, which supports animation of state changes while the interactor is executed.

3. MAIN FUNCTIONALITY

The second, third and sixth step in the development process are supported by the pre-existing scxmlgui editor [3]. Figure 2 depicts the main view of the editor. It supports the visual specification of Harel state charts [2]. Therefore a specification can be composed of events, atomic and complex states, initial states and state entry, exit and transition conditions. Since the tool saves the state charts in the standardized W3C SCXML format [8], states and transitions can be associated with backend API calls and conditions can refer to a data model. During application monitoring an arbitrary interactor state chart that is used in a running application can be loaded. As soon as the editor is connected to the MINT platform it receives state changes of the interactor and highlights active states and corresponding state changes.

With our mapping editor, multimodal mappings are specified by a custom flow chart like notation that offers three basic elements: observations, actions and operators. Boxes with rounded edges describe "observations" of state changes. Boxes with sharp edges are used to define actions, which are backend function calls or the triggering of events. A set of observations is connected by an operator that describes a relation between different modes. Following the CARE properties approach [5] elementary relations can specify: complementary usage (C), can assign an observation

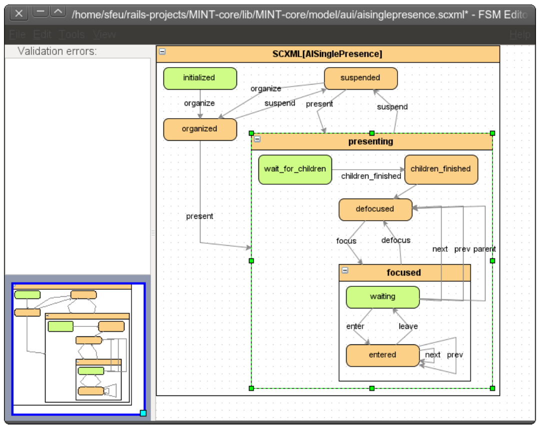


Figure 2. **The main window of the scxmlgui editor.**

[1] http://www.multi-access.de, last accessed August 22th, 2012

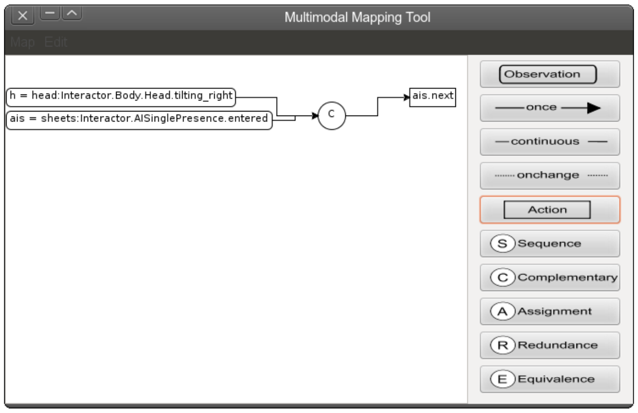


Figure 3. **The main window of the mapping editor.**

to a specific mode (A), can be used redundantly (R) or equivalent (E). Additionally we support the definition of a strict sequence (S) of observations followed by actions. Observations can either be activated instantly as soon as the mapping is started (once), can listen continuously to state changes (onchange) or can combine an initial check for the actual state and a continuous observation till the observation is evaluated to true. Mappings can fail and are then get reinitialized automatically. A comprehensive specification of the mapping language has been published online[2]. Figure 3 depicts a screenshot of the main screen of the tool.

After all interactors and mappings have been specified they are loaded by our runtime environment, the MINT platform, and instantiated as state machines as part of a web server.

Figure 4 shows a screenshot of two interactors running in parallel as state machines: A music sheet widget and a configuration bar to set how to control the page turning: by nodding, tilting the head or by turning it slightly to left or right.

During the application is executed inside the webserver and displayed by a browser, a developer can connect in parallel to the

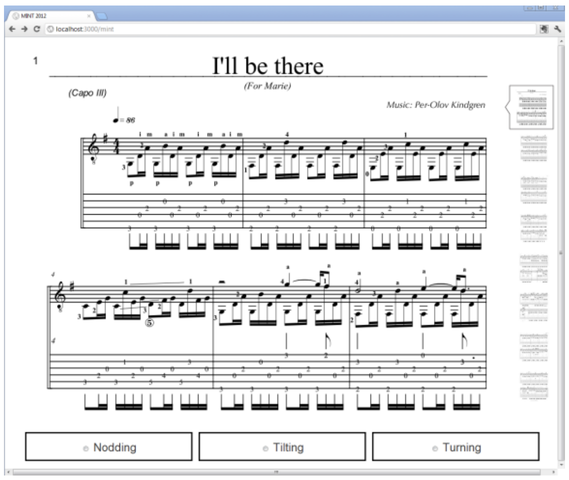
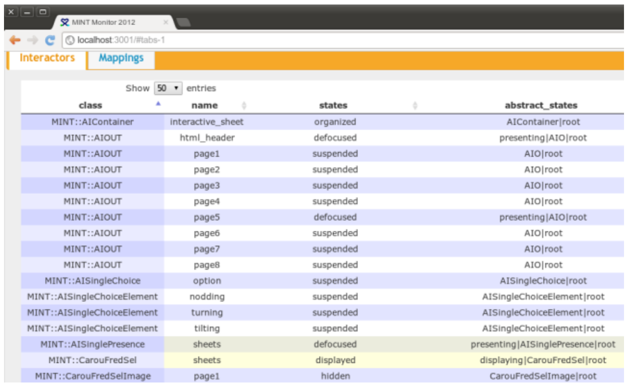


Figure 4. Web browser displaying a music sheets.

[2]http://www.multi-access.de/mint/mim/2012/20120827/, last checked August, 27th, 2012



class	name	states	abstract_states
MINT::AIContainer	interactive_sheet	organized	AIContainer|root
MINT::AIOUT	html_header	defocused	presenting|AIO|root
MINT::AIOUT	page1	suspended	AIO|root
MINT::AIOUT	page2	suspended	AIO|root
MINT::AIOUT	page3	suspended	AIO|root
MINT::AIOUT	page4	suspended	AIO|root
MINT::AIOUT	page5	defocused	presenting|AIO|root
MINT::AIOUT	page6	suspended	AIO|root
MINT::AIOUT	page7	suspended	AIO|root
MINT::AIOUT	page8	suspended	AIO|root
MINT::AISingleChoice	option	suspended	AISingleChoice|root
MINT::AISingleChoiceElement	nodding	suspended	AISingleChoiceElement|root
MINT::AISingleChoiceElement	turning	suspended	AISingleChoiceElement|root
MINT::AISingleChoiceElement	tilting	suspended	AISingleChoiceElement|root
MINT::AISinglePresence	sheets	defocused	presenting|AISinglePresence|root
MINT::CarouFredSel	sheets	displayed	displaying|CarouFredSel|root
MINT::CarouFredSelImage	page1	hidden	CarouFredSelImage|root

Figure 5. The application monitor feature.

monitoring application that tracks all state changes of the interactors in real-time. Figure 5 shows a screenshot of the entire list of interactors running in parallel for the music sheet application. State changes are highlighted in yellow for a short amount of time to ease the observation. A second tab lists in the same manner the running multimodal mappings together with their actual state. The monitor logs all interactions in real-time and serves for (remote) debugging as well as for remote observations of a user interacting with an application.

4. INTERACTIVE MUSIC SHEET

When learning to play a musical instrument, or when playing one, a music sheet is used to give guidance as to how to perform the musical piece. However, as songs become longer and more intricate it may span across several sheets, forcing the player to stop playing to turn the page. Although this may become easier as one becomes more experienced with the instrument, it is a barrier for inexperienced players that can be tackled easily using a different mode to turn the pages other that your hand.

One possibility could be the use of voice to issue a command to turn the page, but this would not be successful as it would disrupt the melody. With these facts in mind we propose a human-computer interface to turn music sheets with simple head movements that can be captured by a basic VGA webcam, a common part of modern notebooks. The considered setup is shown by Figure 6.

We have defined a head movement mode interactor enabling to detect tilting the head to the side, right and left, as the controller

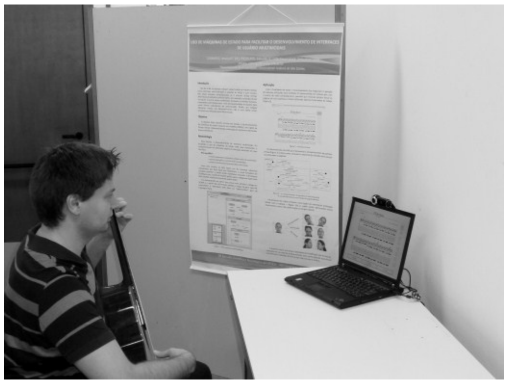


Figure 6. Interacting with the music sheet.

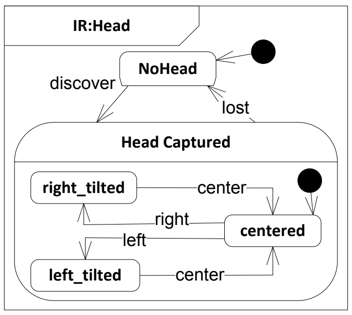


Figure 7. Head interactor, see [3] for the complete specification.

to pass to the next page or return to the previous page, respectively. The corresponding interactor definition is straight-forward like illustrated by figure 7. But to get this interactor working there this still a programming task to be done that links the interactor states to API calls of a head recognition and tracking application.

Figure 8 shows a screenshot from our head tracking application that works well with using a simple VGA camera. The left side of the figure shows the debugging user interface in that a moving arrow represents the direction of the head movement (mirrored).

5. RELATED WORK

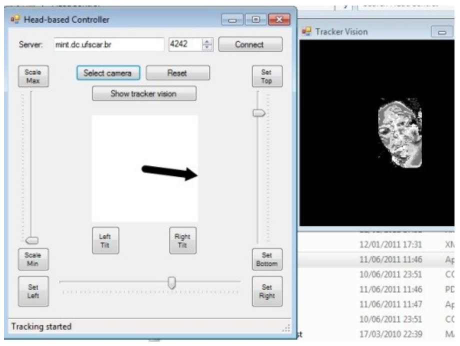
Model-based development to specify and generate user interfaces for several platforms has been practiced since two decades [6]. MBUID is mostly performed to semi-automatically generate interfaces to run on different platforms like a TV, a smart phone or to support different control modes. The generation of multimodal interfaces that consider more than one modality has been only performed to a limited extend [7] to consider XHTML+Voice based web applications. The open interface framework [3] helps assembling multimodal interfaces out of pre-defined components but does not consider a behavior modeling.

6. CONCLUSIONS

This approach using interactors and mappings decreases the work and effort necessary to modify or even add new interaction styles to an already finished interface. For instance, if decided that the style of interaction defined isn't ideal it can be modified or a new one can be designed (e.g., by moving his head from left to right) by simply changing the behavior of the interactors and mappings.

We initially thought the music sheet application just as a small demonstration application to demonstrate what is possible with the proposed tool chain and to explain how easy the interaction can be monitored and changed using our tools. But interestingly controlling page turns using head movements (especially the tilting variant) figured out to work very well, which we initially didn't expected. The reason for this lies in the fact that especially for the case that the music is unknown to the musician she really remains the head in a quite static position while concentrating on "reading" while playing. Only in cases in that the musician did know the music to play very well she focuses more on expressing emotions that actual reading the sheet. But in these cases the sheet itself is no longer required.

[3] http://www.multi-access.de/mint/irm/2012/20120822/, last checked August, 22th



Tracking started

Figure 8. Head Tracking Software.

ACKNOWLEDGMENTS

Sebastian Feuerstack would like to thank FAPESP for funding the travel grant that enabled him to attend the WebMedia 2012 conference and present the paper and is grateful to the Deutsche Forschungsgemeinschaft (DFG) for the financial support of his research.

REFERENCES

[1] Feuerstack S; Colnago J. and De Souza, C.; Designing and Executing Multimodal Interfaces for the Web based on State Chart XML. Proceedings of 3a. Conferência Web W3C Brasil 2011, The 3rd W3C Brazil Web Conference, 17th-18th November, Rio de Janeiro, Brazil.

[2] Harel, D. & Politi, M. (1998) Modeling Reactive Systems with Statecharts: The STATEMATE Approach. New York: McGraw-Hill.

[3] Lawson, J.-Y. L.; Al-Akkad, A.-A.; Vanderdonckt, J. & Macq, B. (2009), An open source workbench for prototyping multimodal interactions based on off-the-shelf heterogeneous components, *in* 'Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems', ACM, New York, NY, USA, pp. 245--254.

[4] Morbini, Fabrizio . Scxmlgui. (Checked, July, 27th 2012) http://code.google.com/p/scxmlgui/

[5] Nigay, L & Coutaz, J. (1997). Multimedia Interfaces: Research and Applications - Chapter 9: Multifeature Systemsn: The CARE Properties and Their Impact on Software Design. AAAI Press.

[6] Puerta, A. R.; Eriksson, H.; Gennari, J. H. & Musen, M. A. (1994), Model-based automated generation of user interfaces, *in* 'AAAI '94: Proceedings of the twelfth national conference on Artificial intelligence (vol. 1)', American Association for Artificial Intelligence, Menlo Park, CA, USA, pp. 471--477.

[7] Stanciulescu, A.; Limbourg, Q.; Vanderdonckt, J.; Michotte, B. & Montero, F. (2005), A transformational approach for multimodal web user interfaces based on UsiXML, *in* 'ICMI '05: Proceedings of the 7th international conference on Multimodal interfaces', ACM Press, New York, NY, USA, pp. 259--266.

[8] State chart xml (scxml): State machine notation for control abstraction, w3c working draft, checked July 27th, 2012 http://www.w3.org/tr/2011/wd-scxml-20110426/