Electronic Communications of the EASST
Volume 45 (2011)

Proceedings of the Fourth International Workshop on Formal Methods for Interactive Systems (FMIS 2011)

Supporting Mobile Application Development with Model-Driven Emulation

Judy Bowen and Annika Hinze

5 pages

Guest Editors: Judy Bowen, Steve Reeves
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Supporting Mobile Application Development with Model-Driven Emulation

Judy Bowen [1] **and Annika Hinze** [2]

Computer Science Department
The University of Waikato, Hamilton,
New Zealand
[1]jbowen@cs.waikato.ac.nz,
[2]hinze@cs.waikato.ac.nz

Abstract: In this paper we outline our proposed approach for supporting mobile application development by using models as inputs to an emulator. We describe a new user interface to a parser tool previously developed, which enables developers to make design decisions about the user interface of the mobile application whilst ensuring that the resulting application adheres to the models. This paper describes work currently in progress.

Keywords: Design, mobile applications, human factors, reliability

1 Introduction

Applications for smart phones have many complex (and often conflicting) requirements and software development for such devices typically takes place using emulators. Smart phone applications provide multiple modes of interaction and feature-rich interfaces on a small screen and may make use of several interacting services and components to provide functionality. Such complex applications with multiple interactions require careful modelling to ensure correctness of both component behaviours and the interactions between those components. When we also take into consideration how users will interact with such systems we require a method of including this in the models, as well as ensuring that the modelled system is correctly implemented. In previous work [HBWM10] we introduced a model-driven emulator for the interaction and GUI design of mobile applications. This took the approach of modelling functional behaviours and user requirements and parsing these in conjunction with design constraints to automatically generate an emulated application. We also discussed how such an approach might be modified to support an interface design-driven process. In this paper we present our initial work on developing such a design-driven process and show the extensions we propose to the parser tool to support this.

2 Introduction to the Example and Process

For this paper we use a simplified example from the design of a tourist information system TIP [HVB09], which has been previously well studied. In this small example the system displays a map to the user along with a reference on the map to their current location (determined via GPS). The user enters the name of a destination to the system and the map is updated to show

Figure 1: Prototypes for the example user interface

the destination along with a marked route from their current location to that destination. The prototypes for the GUI of this example are given in Figure 1. The intention is that once the prototypes have been developed (in conjunction with users and based on their requirements) they are transformed into formal models, namely presentation models and presentation interaction models (PIMs) [BR07]. These models enable us to formally describe the intended behaviours of a prototype (via the presentation model) as well as the navigational possibilities (via the PIM).

At the same time a formal specification of the functionality of the system is developed. In earlier work on the TIP system formal model, discrete event system models (also called VALID models) [HMM06] and UPPAAL [HME09] were used to describe functionality and component interaction. In [HBWM10] we transformed these into μcharts [Ree05] which enabled us to combine them with the user interface (UI) design models which also use μcharts. μcharts describe reactive systems and can be composed with other μcharts enabling us to take UI models and functional models and combine them to produce a model of the entire system.

The first stage in modelling the UI designs is to transform the prototypes into a presentation model (a manual process supported by the PIMed tool [PIM]). The presentation model describes the behaviour of the prototype in terms of its component widgets which consist of a tuple being the widget name, the widget category and the associated behaviours:

(widget name, widget category, (I behaviour, S behaviour)).

Behaviours fall into two categories, *I* behaviours relate to the behaviour of the interface itself (and are, therefore, typically navigation behaviours from one state of the UI to another) and *S* behaviours relate to the underlying functionality of the system (*i.e.* they allow users to perform functional operations). The presentation model for the prototype shown in Figure 1 is given below:

```
ParserGUI is      LocationWin : RouteWin
LocationWin is    (Map, Responder, (S_ShowLocation))
                  (DestEntry, Entry, ())
                  (GoButton, ActionControl, (I_GoRoute, S_ShowRoute))

RouteWin is       (Map, Responder, (S_ShowRoute))
                  (NewButton, ActionControl, (I_GoLocWin, S_ShowLocation))
```

From the presentation model we can then develop the presentation interaction model (PIM), which shows the availability of the behaviours described in the presentation model. Each component of the presentation model (*e.g.* LocationWin and RouteWin) is abstracted as a state in a finite state automata, and transitions between states are added labelled by the I_behaviours of the respective presentation model which represent such a state change. The PIM for the above example is shown in Figure 2.

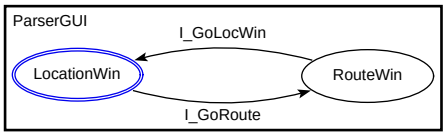

Figure 2: Presentation Interaction Model (PIM)

The existing parser tool reads in a textual description of μcharts (user requirement models and functional models) and creates the necessary Java and XML files to automatically generate an Android emulator application. The intention for the new process is that the parser will also use the PIM and an XML description of the presentation model which will provide the detail of the user interface. Figure 3 shows the proposed process.

3 Additional Requirements for the Parser

Currently when the emulator application is created all of its widgets are determined by the parsed user requirements model, and they are all are naively implemented as buttons. So our previous example would result in an emulator application consisting of two screens each with a single button which would enable the user to toggle between the two screens. What we would like to do is use the additional information from the presentation model and PIM to create a more realistic emulation (*i.e.* one that more closely represents the prototype in both appearance and behaviour). By using the presentation model and PIM in conjunction with the existing models we can ensure that the emulation remains consistant with the formal description and prototype, and therefore, hopefully ensure the final implementation is likewise consistent and correct.

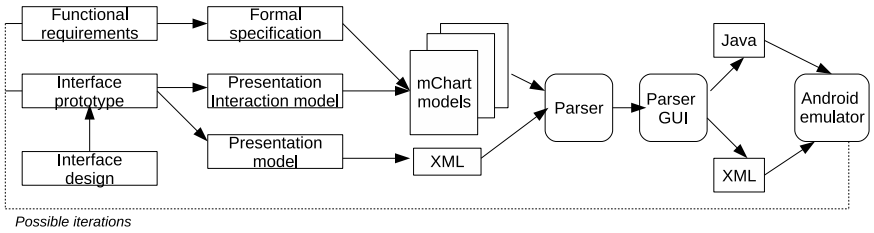

Figure 3: Model-Driven Mobile Simulator: Architecture

3.1 Using the Presentation Model as Input

In order to use the presentation model data as an input to the parser we need to export the model in an XML format and update the parser to include this information. We already have a defined XML DTD for presentation models but will need to extend the PIMed tool to export models in this format (currently it only exports tests in XML). The presentation model data will enable us to populate a front-end GUI for the parser with the names and behaviours of all widgets in the model and enable the developer to select appropriate emulator widgets.

3.2 Parser GUI

The parser GUI is invoked once the developer has imported a set of models to be parsed. We have developed a first version of this which lists widgets (by state) that have associated I_behaviours (*i.e.* those which cause transitions in the PIM) only. Once the parser has been extended to take the presentation model data as an input the parser GUI can be extended so that all widgets will be listed. Figure 4 shows a mockup of how the parser GUI will look once this step has been completed.

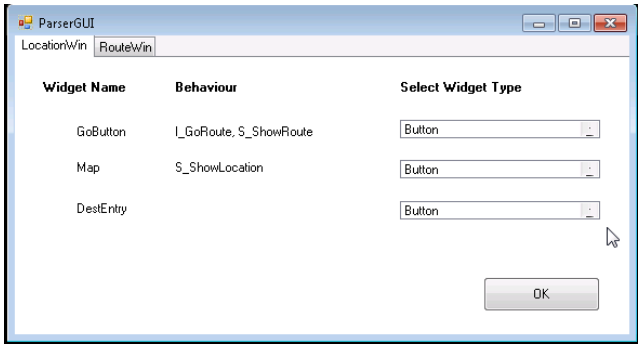

Figure 4: Mock up of parser GUI

We list the I_Behaviour widgets first along with their respective behaviours. The instantiated widget type is set to a default value of **Button** as this is commonly the way such controls will be implemented (the option to change this is still available from the drop-down menu). Then the rest of the widgets and their behaviours are listed with a drop-down menu of available widgets to choose from. As well as all possible widgets that the emulator can support (such as checkboxes, radio buttons, text fields *etc.*) we provide an option **Other** which can be assigned to widgets such as the map. There is no automated way of adding such a widget to the emulator so by selecting the **Other** option we can have a place-holder in the UI until the functional map widget can be coded. In future work it might be possible to link the choice of **Other** to some mechanism whereby the coding for such widgets can be added at this point from within the parser GUI, but that is beyond the aims of the current project.

3.3 Current Status

The parser GUI has been developed in a limited format (as described in the previous section) and is able to correctly add selected widgets to the emulated application. The next step in the project is to change the parser to also accept the presentation model XML in order to display all of the widgets in the GUI and make these selectable. This will also require changes to be made to the PIMed tool to enable the models to be exported in the XML format.

Once these two stages have been completed we will be able to begin testing the parser and GUI on larger sections of the example from the tourist information system TIP so that we can begin gathering feedback on the usability of the tool and investigate other development options which might be useful to support (such as sizing and positioning of widgets).

4 Conclusion

This work is still its early stages, but the development of the initial version of the parser GUI and the experiments performed with this so far suggest that this will be a useful extension to our previous work. Not only will it enable us to develop more realistic applications on the emulator (which in turn enables us to use such emulations for partial user studies) but it ensures that the applications meet the requirements expressed in the formal models without the developer being required to directly consider these.

Bibliography

[BR07] J. Bowen, S. Reeves. Formal Models for Informal GUI Designs. *Electronic Notes in Theoretical Computer Science* 183:57–72, 2007.

[HBWM10] A. Hinze, J. Bowen, Y. Wang, R. Malik. Model-driven GUI & interaction design using emulation. In Sukaviriya et al. (eds.), *EICS*. Pp. 273–278. ACM, 2010.

[HME09] A. Hinze, Y. Michel, L. Eschner. Event-based Communication for Location-based Service Collaboration. In Bouguettaya and Lin (eds.), *ADC*. CRPIT 92, pp. 127–136. Australian Computer Society, 2009.

[HMM06] A. Hinze, P. Malik, R. Malik. Interaction design for a mobile context-aware system using discrete event modelling. In Estivill-Castro and Dobbie (eds.), *ACSC*. CRPIT 48, pp. 257–266. Australian Computer Society, 2006.

[HVB09] A. Hinze, A. Voisard, G. Buchanan. Tip: Personalizing Information Delivery in a Tourist Information System. *Journal of IT & Tourism* 11(3):247–264, 2009.

[PIM] PIMed. An editor for presentation models and presentation interaction models. Available from: http://www.cs.waikato.ac.nz/Research/fm/PIMed.html.

[Ree05] G. Reeve. *A Refinement Theory for μCharts*. PhD thesis, The University of Waikato, 2005.