Semana de Engenharia 2010
Guimarães, 11 a 15 de Outubro

GUIsurfer: A TOOL FOR GUI REASONING

João Carlos Cardoso Silva
Dep. Informática / CCTC, Universidade do Minho, Braga, Portugal
jcsilva.mail@gmail.com

KEYWORDS
GUI, Reverse Engineering, Models

ABSTRACT
The design of interactive systems does not seem to be much improved by the use of tool support. Interfaces are often difficult to understand and use for end users. Moreover, the code produced by available tools is difficult to understand and maintain. In the context of an effort to develop tools to support the automated analysis of interactive system designs, we are developing GUIsurfer, a tool capable of deriving and reasoning about models of graphical user interfaces from source code. The goal is to enable the analysis of existing interactive applications, for example, when they must be ported or updated.

INTRODUCTION
In order for an user interface to have good usability characteristics it must both be adequately designed and adequately implemented. Tools are currently available to developers that allow for fast development of user interfaces with graphical components. However, the design of interactive systems does not seem to be much improved by the use of such tools. Interfaces are often difficult to understand and use for end users. In many cases users have problems in identifying all the supported tasks of a system, or in understanding how to achieve their goals (Loer et al., 2005). Moreover, these tools produce spaghetti code, which is difficult to understand and maintain (Mikkonen and Taivalsaari, 2008). The generated code is composed by call-back procedures for most widgets like buttons, scroll bars, menu items, and other widgets in the interface. These procedures are called by the system when the user interacts with the system through an widget's event. Graphical user interfaces may contains hundreds of widgets, and therefore many call-back procedures which turn difficult to understand and maintain the source code. Additionally, it is important to ensure that Graphical User Interfaces (GUI) based applications behave as expected. The correctness of the GUI is essential to the correct execution of the software.

Model-based development and formal methods could be used to validate system requirements early in the life cycle at reasonable cost. Different types of models can be used in the design and development of interactive systems, from user and task models to software engineering models of the implementation (Blandford, 2004).
In the context of an effort to develop tools to support the automated analysis of interactive system designs, we are investigating the applicability of reverse engineering approaches to graphical user interface (GUI) analysis from source code. Our objective consists in developing tools to automatically extract models of the GUI structure and behaviour, from its source code. The models should specify which widgets are present in the interface, and their relationships; which particular GUI events can occur, and when; which are the conditions affecting the effect of those events in the interface; which system actions are executed; and which GUI state is generated next.

We want to be able to reason and test these GUI models in order to analyse aspects of the original application's behaviour, and the quality of the implementation. To that end, we are developing GUIsurfer, a tool capable of automatically deriving and reasoning about structural and behavioural GUI models of applications written in Java/Swing, WxHaskell, or GWT.

This work enables the analysis of existing interactive applications, for example, when an existing application must be ported or simply updated (Moore, 1996). Being able to reason at a higher level of abstraction than that of code will help in guaranteeing that the new/updated user interface has the same characteristics of the previous one.

THE ARCHITECTURE OF GUIsurfer

The GUIsurfer tool is composed by four modules (see Figure 1): FileParser, AstAnalyser, GuiModelAnalysis and Graph. These modules are configurable through command line parameters. Below we outline some of the more important functionalities for each module.

- FileParser - Currently this module allows the parsing of Java/Swing or WxHaskell source code. The FileParser tool is language dependent.
- AstAnalyser - This module is used to extract the GUI layer from any abstract syntax tree. This process is parameterized with constructors. The AstAnalyser tool is another language dependent tool used to slice an abstract syntax tree, considering only it graphical user interface layer. Part of this tool is easily retargetable, however most of the tool needs to be rewritten to consider another particular programming language. The AstAnalyser tool is composed of a slicing library, containing a generic set of traversal functions that traverse any AST. This tool must be used with three arguments, i.e. the abstract syntax tree, the entry point in source code

(e.g., the main method for Java source code), and a list with all widgets to consider during the GUI slicing process.

- GuiModelAnalysis - This module implements the GUI abstraction step. From several fragments of the GUI layer, this module automatically generates a Haskell GUI's specification and other GUI models;
- Graph - Finally the Graph module is used to test the graphical user interface through it's Haskell GUI's specification. To implement this task we make use of QUICKCHECK. The Graph tool is language independent and generates several metadata files with events, conditions, actions, and states extracted form source code. Each of these types of data is related to a particular fragment from the AST. Another important output generated by the Graph tool are GUI specifications written in the Haskell programming language. These specifications define the GUI layer mapping events/conditions to actions. This tool allows us also to generate several visual models through the GraphViz tool, such as state machines, behavioral graph, etc.

CONCLUSIONS AND FUTURE WORK

The current version of GUIsurfer has been developed to reverse engineering interactive applications from source code. This work has shown that model-based reasoning provides an easy way to implement interactive systems analysis. Models provide a tool to explore GUI properties. We evaluated these models from different case studies, which demonstrated how model-based reasoning enables us to test interactive systems.

In what concerns user interface development, two perspectives on quality can be considered. Users, on the one hand, are typically interested on what can be called external quality: the quality of the interaction between users and system. Programmers, on the other hand, are typically more focused on the quality attributes of the code being produced.

This work is an approach to bridging this gap by allowing us to reason about GUI models from source code. We described GUI models extracted automatically from the code, and presented a methodology to reason about the user interface model. A number of metrics over the graphs representing the user interface are being investigated. Some ideas on testing the graph against desirable properties of the interface are also being explored.

ACKNOWLEDGMENT

This work is supported by Fundação para a Ciência e Tecnologia (FCT, Portugal) through PhD Grant SFRH/BD/1179/2006.

REFERENCES

A. Blandford, R. Butterworth, and P. Curzon. Models of interactive systems: a case study on programmable user modelling. International Journal of Human-Computer Studies International Journal of Human-Computer Studies, 60:149–200, 2004.

K. Loer and M.D. Harrison. Analysing user confusion in context aware mobile applications. In M.F. Constabile and F. Paternó, editors, PINTERACT 2005, volume 3585 of Lecture Notes in Computer Science, page 184-197, New York, NY, USA, 2005. Springer.

T. Mikkonen and A. Taivalsaari. Web applications - spaghetti code for the 21st century. In International Conference on Software Engineering Research, Management and Applications, pp. 319–328, 2008.

M. Moore. A survey of representations for recovering user interface specifications for reengineering. Technical report, Institute of Technology, Atlanta GA 30332-0280, june 1996.

AUTHOR BIOGRAPHY

JOÃO CARLOS C. SILVA

Licentiate in Computer Science and Mathematics. He is currently doing a PhD at University of Minho with the thesis title: *GUI SURFER: A Tool for Reverse Engineering of Graphical User Interface.*