Using Model Checking to Generate Test Cases for Android Applications

Ana Rosario Espada   María del Mar Gallardo   Alberto Salmerón   Pedro Merino

Dept. Lenguajes y Ciencias de la Computación
E.T.S.I. Informática   University of Málaga*

[anarosario,gallardo,salmeron,pedro]@lcc.uma.es

The behavior of mobile devices is highly non deterministic and barely predictable due to the interaction of the user with its applications. In consequence, analyzing the correctness of applications running on a smartphone involves dealing with the complexity of its environment. In this paper, we propose the use of *model-based testing* to describe the potential behaviors of users interacting with mobile applications. These behaviors are modeled by composing specially-designed state machines. These composed state machines can be exhaustively explored using a model checking tool to automatically generate all possible user interactions. Each generated trace model checker can be interpreted as a test case to drive a runtime analysis of actual applications. We have implemented a tool that follows the proposed methodology to analyze ANDROID devices using the model checker SPIN as the exhaustive generator of test cases.

1 Extended abstract

Our model-based testing [5] proposal is based on the idea that each application is associated with a set of *intended user behaviors*, instead of being used randomly. For each application screen, we use state machines to construct a *non deterministic model* which represents the expected use of the screen/application. This state machine is built semiautomatically, with information provided by the expert (the app designer or tester) and by ANDROID tools like UIAUTOMATORVIEWER. Then, all these view models may be conveniently composed to construct a *non deterministic* model of the user interaction with a subset of mobile applications of interest. Each execution trace of the *composed state machine* corresponds to a possible realistic use of the mobile. Thus, the generation of test cases is reduced to the generation of all possible behaviors of the composed machine, which may be carried out by a model checking tool.

We define the behaviour of mobile applications through the composition of state machines at different abstraction levels. The lowest level is composed of *view state machines*, each one corresponding to an app screen. The view contains the controls (e.g. buttons, text fields) through which the user interacts with the app, and which are used as events to trigger transitions. Only one view may be active at the same time, and thus be manipulated by the user. Some user events may change the view that is currently active. We have modeled this control transfer between views through the *composition relation* of view state machines from which *device state machines* are constructed. Device state machines use the *connection states* to switch from the current active view to a different view.

We use the SPIN model checker [4] in our approach for automatically generating test cases from application models in the following way. First, each device will be represented by a single PROMELA process, which contains a state machine that models the applications contained on that device. The state

*Work partially supported by grants P11-TIC-07659 (Regional Government of Andalusia), TIN2012-35669 (Spanish Ministry of Economy and Competitiveness), UMA-Agilent Technologies 808/47.3868- 4Green and the AUIP as sponsor of the Scholarship Program Academic Mobility.

Submitted to:
PROLE 2015

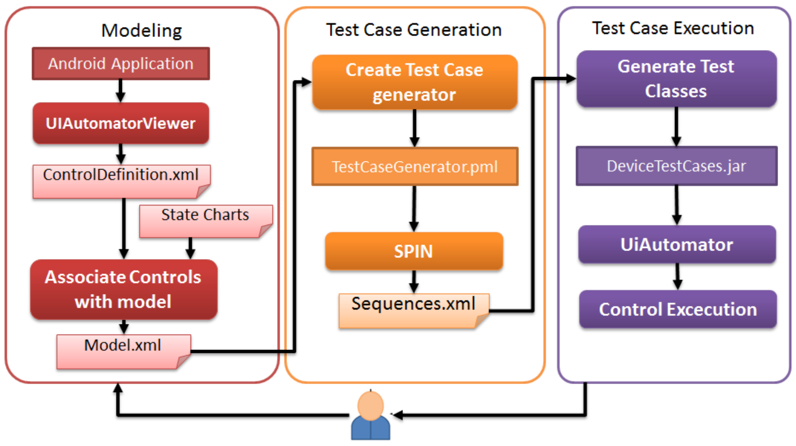

Figure 1: Test generation and execution process

machines themselves are written as loops, where each branch corresponds to a transition triggered by an event. The current state of each state machine (stored as a global PROMELA variable) determines which branches are active and may be taken. This PROMELA specification is explored exhaustively by SPIN in order to generate all possible test cases described by the application model, taking all possible alternatives when there is more than one active branch at the same time. Finally, the test cases are translated into JAVA classes which carry out the user actions in the ANDROID device.

The paper provides two main contributions. The first one is the formal definition of a modular approach for modeling user interaction with mobile applications using state machines. The second one is a method to employ the explicit model checker SPIN [4] to generate test cases by exploring the composed state machine. We have constructed a tool chain which implements both modeling and test generation phases to shows the feasibility of the approach in practice, summarized in Figure 1. The next step would be analyzing the execution of these test cases using a runtime verification module [3], which is currently being addressed by the authors of this paper [1].

This paper is an extended abstract of [2], presented at the 10th Workshop on Model Based Testing.

References

[1] Ana Rosario Espada, María-Mar Gallardo, Alberto Salmerón & Pedro Merino (2015): *Runtime verification of expected energy consumption in smartphones*. In: *Proceedings 22nd International Spin Workshop on Model Checking Software*, *LNCS* to appear, Springer.

[2] Ana Rosario Espada, María-Mar Gallardo, Alberto Salmerón & Pedro Merino (2015): *Using Model Checking to Generate Test Cases for Android Applications*. In Nikolay Pakulin, Alexander K. Petrenko & Bernd-Holger Schlingloff, editors: Proceedings Tenth Workshop on *Model Based Testing*, London, UK, 18th April 2015, *EPTCS* 180, Open Publishing Association, pp. 7–21, doi:10.4204/EPTCS.180.1.

[3] Ana Rosario Espada, María-del-Mar Gallardo & Damián Adalid (2013): *A Runtime Verification Framework for android Applications*. In: *Proceedings of XXI JCSD*.

[4] Gerard J. Holzmann (2003): *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley Professional.

[5] Mark Utting & Bruno Legeard (2007): *Practical Model-Based Testing: A Tools Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.