TouchGuru: Integrating Static Analysis with a Mobile Development Environment

Lucas Brutschy
Department of Computer Science
ETH Zurich
lucas.brutschy@inf.ethz.ch

Pietro Ferrara
IBM Thomas J. Watson
Research Center
pietroferrara@us.ibm.com

Peter Müller
Department of Computer Science
ETH Zurich
peter.mueller@inf.ethz.ch

Abstract

Mobile apps often expose bugs only under specific environment conditions or after specific interactions, making it hard to detect them via testing. By approximating the program behavior under all possible conditions, static analysis is able to fill this gap. In this invited talk, we present TouchGuru, a static analyzer for Microsoft TouchDevelop scripts. In particular, we present how the design of TouchGuru takes into account the user's perspective.

1. Introduction

TouchDevelop [2] is a novel programming environment and language to develop mobile apps on mobile devices. These so-called scripts are typically developed by non-expert users, rather small, and published in the cloud. The short script in Figure 1 defines an event handler that is executed each time the mobile device is shaken. The body of the event handler selects and plays a random song from the media library of the device.

The development of mobile applications is challenging, because they might be executed on diverse platforms, access a volatile environment and are driven by an unpredictable sequence of events. This makes testing challenging especially for non-professional developers, who did not receive formal training and do not have access to testing infrastructures.

Static semantic program analysis provides a viable solution to this problem: It is automatic, requiring only minimal interaction with the user, and it can model the behavior of the program under all viable execution conditions to detect reliability issues. In this talk we present TouchGuru [1], a static analysis for TouchDevelop scripts.

MobileDeLi '14, October 21, 2014, Portland, OR, USA.

ACM 978-1-4503-2190-7/14/10.
http://dx.doi.org/10.1145/2688412.2688421

```
event shake ()
  media → songs → random → play
```

Figure 1. Playing a random song of the media library

For instance, the example in Figure 1 contains a reliability issue exposed only when executed on devices with an empty song list. In this case, `random` will return an *invalid* value, causing the script to crash when `play` is executed. TouchGuru reports potential runtime errors like this one. Other examples of the alarms reported are incorrect initialization of values, not verifying the existence of certain hardware peripherals before accessing them, or simply accessing a collection out-of-bounds.

Traditionally, there are three main properties that researchers consider when developing a static analyzer: (1) soundness, that is, no real violation of the checked property is missed, (2) precision, that is, few or zero alarms are not real violations, and (3) efficiency, that is, the analysis terminates quickly. However, users may have slightly different requirements. In particular, efficiency for them means also that the analysis can be *used* efficiently, e.g., that the alarms can be understood and fixed quickly. Similarly for soundness, the analysis should report the alarms they care about, and they can fix.

In this talk, we present three core design choices we made to specifically take into account the user's perspective. Our previous work [1] presents the technical details and experimental results of the analysis.

2. Late Failing and Error Reporting

When a script performs an infeasible action, such as accessing an unavailable sensor, TouchDevelop APIs do not abort execution but instead return an invalid value. The execution is aborted only if the invalid value is used as receiver or parameter of a method call. We call this behavior *late failing*. TouchGuru has to reflect late failing in its analysis methodology: An operation that may fail and return an invalid value is not reported as an error. However, once an invalid value is used in an inappropriate context, an alarm is emitted.

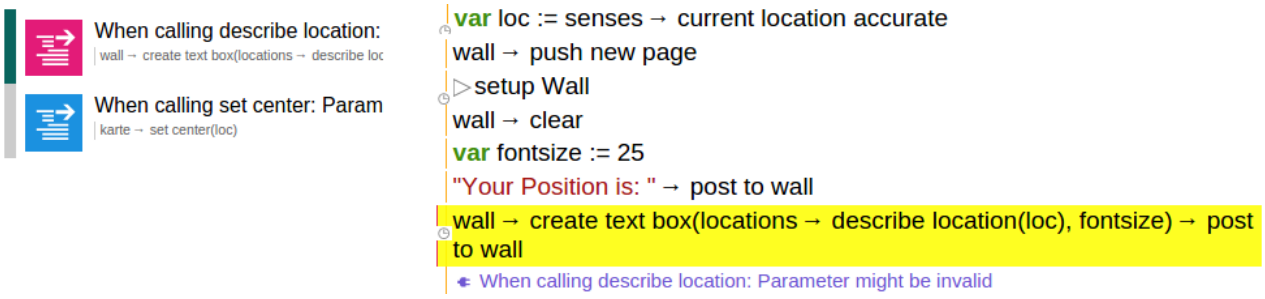

Figure 2. TouchGuru User Interface. The list of all alarms is shown on the left. The yellow box on the right indicates one of them, an illegal use of an invalid value.

Consider again the script in Figure 1. The program does not check if `media->songs` is empty before calling `random`, and therefore `random` might produce an invalid value. However, the program execution continues until the invalid value is used as the receiver of the `play` method.

In this minimal example, the cause and the effect happen to be in the same line; however, in longer programs, the cause is typically located different lines, functions or libraries, which makes it hard for the user to understand and fix the alarm. To address this problem, TouchGuru tracks the source of invalid values by annotating each abstract invalid value with the program point that generates it as well as a textual cause description. This information is then propagated to the location of the alarm, allowing the user to immediately jump to the origin of the invalid value and fix it.

3. Environment and Error Model

Assume the user fixes the bug from Figure 1 by introducing a check whether the media library contains at least one song:

```
if media → songs → count > 0 then
    media → songs → random → play
```

This change seems to fix the problem and the user might expect the alarm to disappear. However, the program could still crash: The media library is shared with other applications on the device, and it may be emptied between the first and the second line in the example above. Therefore, a strictly sound static analysis would still report the alarm. However, since the script does not have exclusive access to the media collection of the device, the code cannot be fixed. Therefore, it is useful to report it.

We have developed an environment model for TouchGuru that finds the right balance between soundness and usefulness of the analysis. For example, we assume that the music library does not change during the execution of a single action or event handler, but may change in between. This choice is based on the intuition that a script may run for a long time, but actions and event handlers typically terminate quickly, making media library changes during an action or event handler unlikely. Under this assumption, no alarm has to be emitted for the above example.

4. Cloud Integration

The integration of TouchGuru into the development environment is crucial for allowing efficient use. The TouchDevelop IDE runs as a web application in the cloud, and is thereby constantly connected with a variety of services. TouchGuru is integrated directly into this environment as an installation-free plugin.

The TouchGuru plugin is integrated into the web application running on the programmer's device, and communicates with a dedicated analysis server based at ETH Zurich. When a user requests the analysis of a script, the code is sent to the analysis server together with a unique ID. The analysis server acquires the required libraries from the TouchDevelop cloud and starts the analysis. When analysis terminates, the results are sent back to the TouchGuru plugin and displayed to the user. Thanks to the dedicated server and the strong integration with the IDE, the user can efficiently run the analysis and access its results.

5. Accessing TouchGuru

Figure 2 shows a screenshot of TouchGuru. TouchGuru is accessible to any TouchDevelop user. You can find TouchDevelop at `http://www.touchdevelop.com`. To make use of TouchGuru, select "Plugins" while editing a script and search for "TouchGuru". Click on the corresponding search result and press "Run analysis". When the analysis terminates successfully, a list of all alarms are displayed on the left. Clicking on one of the alarms takes the user to the location of the potential program crash. From here, additional buttons on the bottom of the screen can be used to highlight the cause of the alarm, or ignore the error message.

References

[1] L. Brutschy, P. Ferrara, and P. Müller. Static analysis for independent app developers. In *Proceedings of OOPSLA '14*, 2014.

[2] N. Tillmann, M. Moskal, J. de Halleux, and M. Fahndrich. Touchdevelop: Programming cloud-connected mobile devices via touchscreen. In *Proceedings of SPLASH/Onward! '11*.