Using Gherkin for Interaction Design Testing and Monitoring *

Abigail Cauchi[1], Christian Colombo[1], Mark Micallef[1], and Gordon Pace[1]

University of Malta

In software testing, Controlled Natural Languages (CNLs), such as Gherkin, are sometimes used to help clients to communicate with developers and draw up specifications. These Gherkin specifications are then translated to tests. Since testing is not exhaustive, using runtime monitoring on critical systems helps ensure that programs run as they should while deployed. Runtime monitoring is not yet (widely) adopted by industry [3] and in this work, we are looking into automatically translating tests into runtime monitors.

We take the case of safety critical number entry systems design in medical devices. Interaction designers and psychologists, study how to best design these interfaces such that the likelihood of human error is reduced. Non-technical designers then use a CNL to communicate their findings with software developers/testers, who will then draw up unit tests. In this abstract we propose to automatically translate these tests to runtime monitors in order to ensure that the system works correctly when deployed, without any additional cost to the manufacturer. Finally, we show the corresponding automata for the runtime verification tool, LARVA [2].

1 Directional number entry system case study

A directional number entry system displays a value on the screen and a cursor that highlights the selected digit. The up and down buttons change the highlighted digit and the left and right buttons move the cursor. From [1] we see that if the cursor is on an edge digit, for example the leftmost position, if left is pressed, the cursor should remain on the leftmost position and the user should be alerted that the interface has not changed following the key press.

An interaction designer and a software developer discuss the property and draw up a Gherkin specification as follows:

```
Scenario: Going beyond left boundary
Given cursor on leftmost position
 When Left is pressed
 Then Cursor position stays the same
 And Displayed number stays the same
 And User is alerted
```

We can translate this specification to a unit test by first setting the display with a number and setting the cursor on the leftmost position; then store the display and cursor position. We then call the event handler of the left button press and assert that the initial cursor position and the display match the initial settings, and that the user is alerted. This can be seen in the code below.

```
public void leftButtonBoundaryTest()
{ DirectionalPad dPad = new DirectionalPad();

  //Setting the display & cursor for the unit test
  dPad.setDisplay("9000.0");
  dPad.setCursorPosition(1); //cursor on the 9

  //storing the display & cursor
```

* Project GOMTA financed by the Malta Council for Science & Technology through the National Research & Innovation Programme 2013

```
9   string initialDisplay = dPad.getDisplay();
10  int initialCursorPos = dPad.getCursorPos();
11
12  //performing the left action
13  dPad.btnLeft_Clicked(this, null);
14
15  //asserting that the three cases specified in Gherkin are true
16  assert(initialCursorPos == dPad.getCursorPos());
17  assert(initialDisplay == dPad.getDisplay());
18  assert(dPad.isUserAlerted);}
```

The runtime monitor will monitor code at execution time. We start the monitor by getting the current cursor position and display. When the left button is pressed and the position and display remain the same; and the user is alerted, the monitor remains in the OK state. This transition is directly related to the three assert statements in the unit test. Any other outcome from the left button being pressed while the cursor is on the leftmost position, then the monitor goes into a bad state. The LARVA automata can be seen in figure 1.

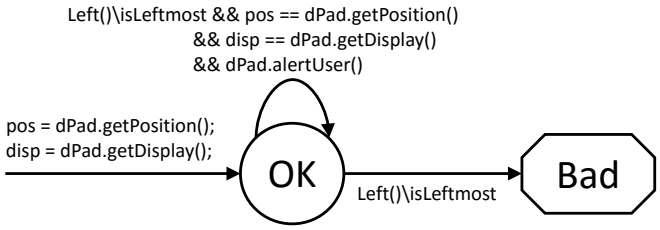

Fig. 1. LARVA Automata

2 Way Forward

It is important to get user interaction design right, especially in safety critical domains. The field of HCI is multidisciplinary — psychologists, designers and computer scientists work together to develop usable and useful software systems. For such a field, a CNL such as Gherkin is useful for bridging the communication gap between psychologists and software developers.

In this abstract we have shown how Gherkin can be used to describe safety critical number entry systems design properties and we discussed how we can unit test these properties and monitor them at runtime. For future work, this process will be completed for the domain of directional number entry systems, then we will look into how it can be generalized.

References

1. A. Cauchi, H. Thimbleby, P. Oladimeji, and M. Harrison. Using medical device logs for improving medical device design. In *Proceedings of the 2013 IEEE International Conference on Healthcare Informatics*, ICHI '13, pages 56–65, Washington, DC, USA, 2013. IEEE Computer Society.
2. C. Colombo, G. J. Pace, and G. Schneider. Larva — safer monitoring of real-time java programs (tool paper). In *Seventh IEEE International Conference on Software Engineering and Formal Methods (SEFM)*, pages 33–37. IEEE Computer Society, November 2009.
3. N. Decker, M. Leucker, and D. Thoma. junitrvadding runtime verification to junit. In G. Brat, N. Rungta, and A. Venet, editors, *NASA Formal Methods*, volume 7871 of *Lecture Notes in Computer Science*, pages 459–464. Springer Berlin Heidelberg, 2013.