Joey Baldwin, Eric Hofesmann, Marge Marshall

CSCI 362 Software Engineering

November 29, 2016

Chapter 5: Fault Injection

Experiences:

The fault injections for our framework were fairly straightforward -- it was easy to identify which tests were going to fail. Injection 1 changed every result for method noteToFreq(), injection 2 changed every result for frange() that was not a thrown exception, injection 3 changed the results for tests of frange() whose stop value was one step above the expected final value, injection 4 changed the results of noteToFreq() tests where the midi value was intended to be rounded up rather than down, and injection 5 changed the results for colorGradient tests that had a change in the red value. An important lesson to be learned from these fault injections is that not every fault can be discovered by every test. There were instances in which a test still passed even though there was an error in that method.

Injected Faults:

music.py:

(1) Tests 0 - 4: noteToFreq, comment line 1239 uncomment line 1240, we changed

frequency = concertPitch * 2 ** ((note - 69) / 12.0)

to

frequency = concertPitch * 2 ** ((note - 70) / 12.0)

Tests 0 - 4: failed

(2) Tests 5 - 10: frange, comment 611 uncomment 612, we changed

if step > 0:

to

if step < 0:

Tests 5 - 9: failed, results only contained the first value in the list

(3) Tests 5 - 10: frange, comment line 614 uncomment line 615 we changed

done = start >= stop

to

done = start > stop

Tests 5, 6: failed, results included an extra value at the end of the list

(4) Tests 16 - 20: freqToNote, comment line 1225 and uncomment line 1226, we changed

note = round(x)

to

note = x

Tests 16 and 20: passed

Tests 17,18, and 19: failed

gui.py:

(5) Tests 11-15: colorGradient, comment line 337 uncomment line 338

differenceR = red2 - red1

to

differenceR = red1 - red2

Tests 12,13: passed

Tests 11,14, and 15: failed