The Full Story – Checking a Big Output

Ken Pugh, ken@kenpugh.com, https://kenpugh.com

In "Building Collaboration with Visible Tests", we saw how to write scenarios for three of the requirements of the FizzBuzz kata. In this article, we'll look at the remaining requirement – a big output and alternative ways to create tests for it using BDD and Approval Testing (Llewellyn Falco and Emily Bache). The same approaches can be used for other applications which have large amounts of output.

First a quick review of the FizzBuzz requirements:

Requirement

The requirement is:

- Write a program that prints one line for each number from 1 to 100
- For multiples of three print "Fizz" instead of the number
- For the multiples of five print "Buzz" instead of the number
- For numbers which are multiples of both three and five print "FizzBuzz" instead of the number

The previous article covered creating scenarios/test for the last three requirements.

An Output Test

One could list the entire output that is desired. This could take a fair amount of time. But it would be a full test.

```
Scenario: FizzBuzz output
* Fizzbuzz output is
| Output    |
| 1         |
| 2         |
| Fizz      |
| 4         |
| Buzz      |
| Fizz      |
| 7         |
| 8         |
| Fizz      |
| Buzz      |
| 11        |
| Fizz      |
| 13        |
| 14        |
| FizzBuzz  |
# and so forth up to
| Buzz      |
```

Testing With an Oracle

An Oracle is a process or another application that produces the correct results. When you are writing your application, you compare your output to the one the oracle creates for the same input. If they do not match, then your implementation has an error. The oracle is "always right".

Having multiple implementations that perform equivalent computations is a standard practice to achieve reliability. If the output from all of them matches, then it is considered correct. If there is one that doesn't match, then the output from the others is used, and a notification sent that there is an issue with the non-matching one. If all the implementations disagree, then it's time for a discussion.

So, another team could implement FizzBuzz and you could compare the outputs. If they matched, you'd have some confidence that you both got it right. If they didn't, then you'll need to have a discussion. If lots of teams independently create a FizzBuzz application and they all match, you'd have a great deal of confidence. FizzBuzz is simple enough that you might not need more than one other application, but if you were navigating to the moon, you might want a few more.

Sampling Tests

You could create a test that checks only a sample of the output. When the potential output is large, this will be a cost/benefit tradeoff. You should choose carefully what parts of the output you want to sample. To do so, you may need to have some "testpoints", which are alternative ways to control the application or to obtain output from it. The testpoints may be controlled by configuration files, a test version of the application, or other means.

For FizzBuzz, you could create output for a sequence specified by a starting and ending number. You could select several sequences, depending on what you think are critical ones.

```
Scenario: FizzBuzz output
Given sequence is
| Minimum | Maximum |
| 4       | 7       |
Then Fizzbuzz output is
| Output    |
| 4         |
| Buzz      |
| Fizz      |
| 7         |
```

Approval Testing

Another way to test the output is to execute the application and save the output. Then the customer or subject matter expert looks at the output and sees if it is correct. If it is, then that output becomes the oracle for comparison on future tests. This is only a small example of approval testing, see https://www.methodsandtools.com/archive/approvaltest.php for more details. If you don't have automated tests for a legacy application that you are going to refactor, it is one way to create them.

In the case of FizzBuzz, the output (or portions of it) would be approved and then used as the test to ensure any implementation changes did not alter the behavior.

Summary

Test driving the entire FizzBuzz application requires having a test for the first requirement – “prints one line for each number from 1 to 100”. We’ve shown four ways of doing this. You may come up with more. You can use the same approach to other applications that need tests for large outputs.