Software Testing, Quality Assurance & Maintenance—Lecture 6

Patrick Lam
University of Waterloo

January 23, 2026

Background: schist—metamorphic rock.

Part I

Metamorphic Testing

No Oracle? Now what?

Want to run zillions of tests.
What's the expected output?

We had implicit oracles, e.g. "doesn't crash".

Can we do better?

Consider min

Suspend disbelief & pretend we have no oracle:

```
def min(a,b):
    if a < b:
        return a
    else:
        return b
```

How can we generate heaps of tests?

“New tests from old”

“New tests from old” II

Say `min(3, 5) = X`.

Can we construct another test case with known output?

“New tests from old” III

`min(5, 3) = X` also.

A simple example of metamorphic testing: even if you don’t know X, it’s the same X:

`min(5, 3) = min(3, 5).`

Always domain-specific.

Observations on metamorphic testing

You are testing f and you have input x_0.

1. given one input x_0, generate another input x_1;
2. you know some property of $f(x_1)$ in relation to $f(x_0)$;
3. you don't necessarily know $f(x_1)$;
4. in fact, you don't need to know $f(x_0)$ in advance!

min and metamorphic testing: no outputs

You can randomly generate 1 zillion test inputs for `min`.

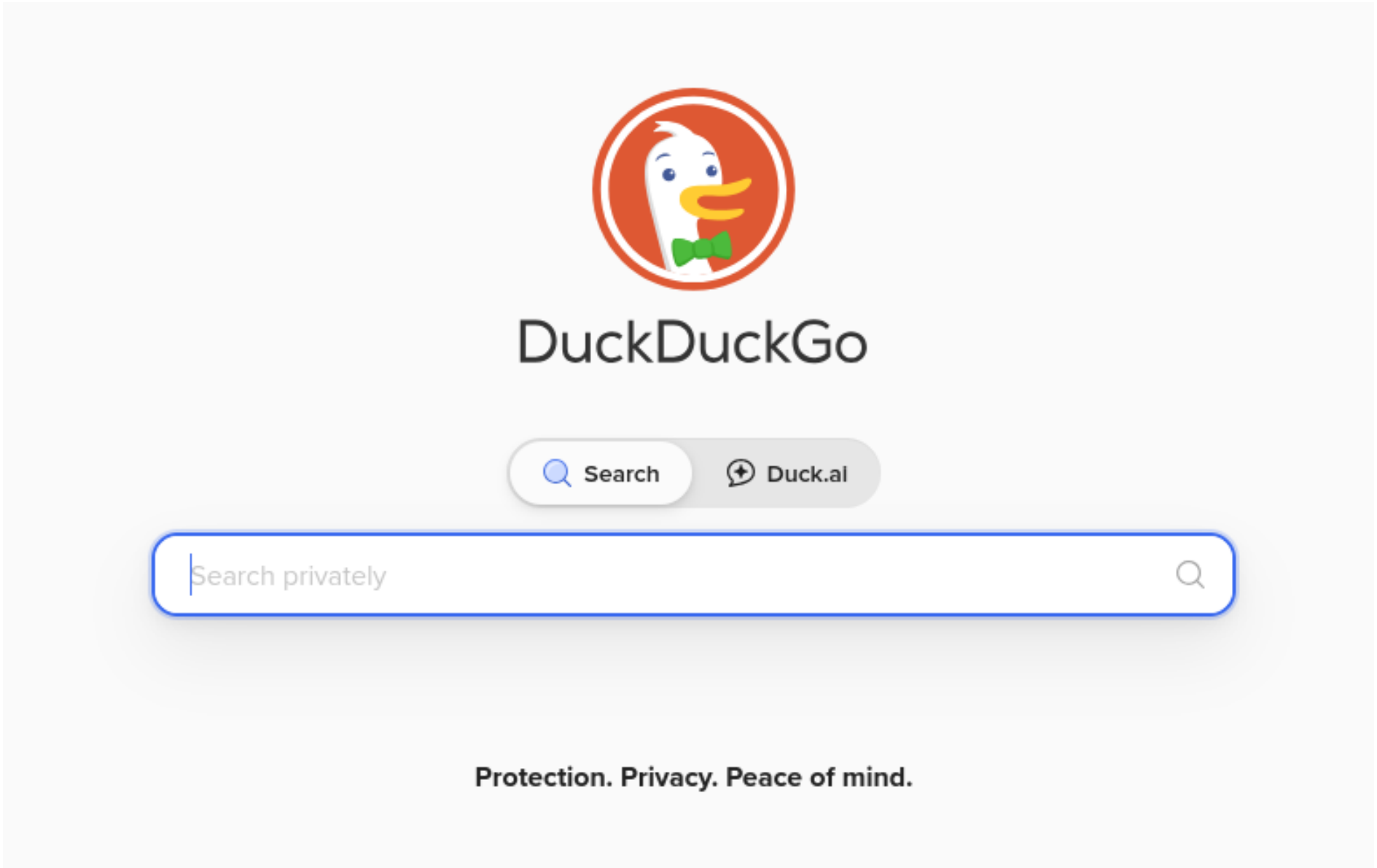
By our assumption, we don't have the corresponding outputs,
just that `min` shouldn't crash.

min and metamorphic testing: what we have

- can generate another 1 zillion test cases, by inverting parameter order.
- also don't have outputs for these cases, but
- we know that it should be the same as the uninverted input!

That's something, which is better than nothing.

Another example: search



Exclusion

1. Search for “metamorphic”.
2. Get, say, 3300 results.
3. Search for “metamorphic -testing
   (i.e. exclude tesing).
4. Now get 4200 results.
5. ???

How do we use this insight to get new tests from old?

Another example: text-to-speech

Speech-to-text scenario

You are writing a speech-to-text processor.

Input: audio file.
Output: text.

Testing speech-to-text

1. manual test generation: you record a text and compare to known good output (oracle).
2. you (or I) get 130 students to do that.
3. you generate audio using text-to-speech.
4. you use Mechanical Turk to get audio.

Still not even close to covering the space of valid inputs, e.g. accents.

Getting lots of inputs. . .

Well, there are a lot of audio files on the Internet. . .

. . . but there is no ground truth, and human oracles are expensive.

What about property testing?

"It doesn't crash on any input." OK...
"It doesn't transcribe acoustic music into words." I guess?

These properties don't really validate the system.

Transforming inputs

Let's say you have one input, which transcribes to `out`. You can:

- double the volume; or,
- raise the pitch; or,
- increase the tempo; or,
- add background static; or,
- add traffic noises; or,
- combine any of these.

Implications of transforming inputs

“add traffic noises” has a lot of freedom;
you can add 10 different traffic noises.

Then, double the volume on each of them;
now have 22 test cases.

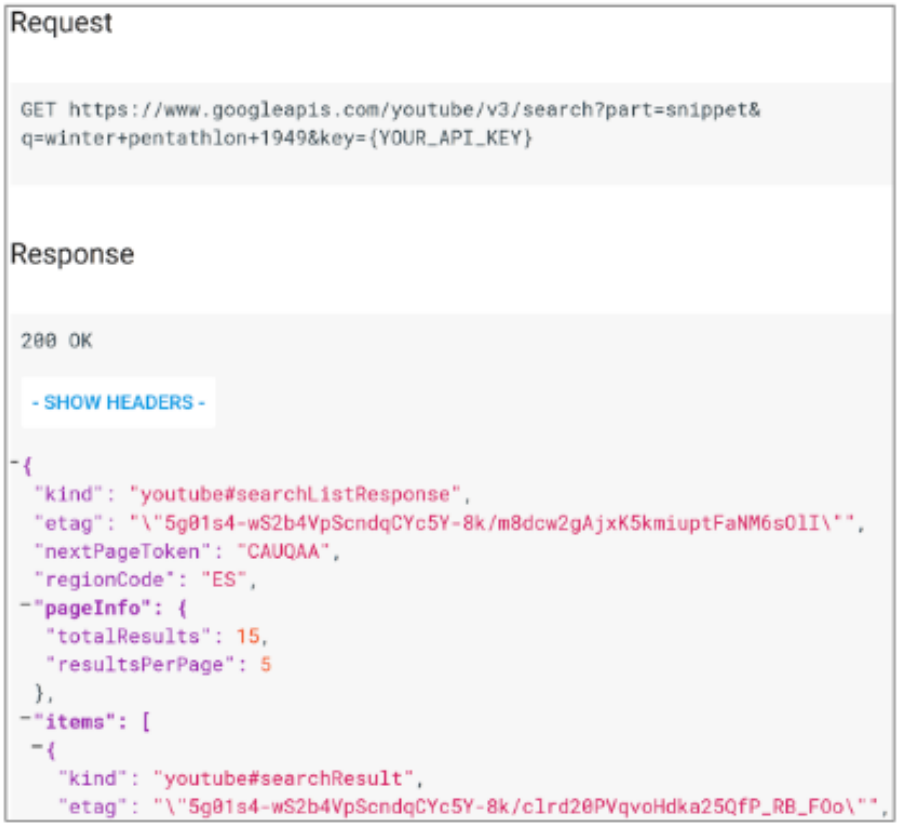
etc.

Further implications

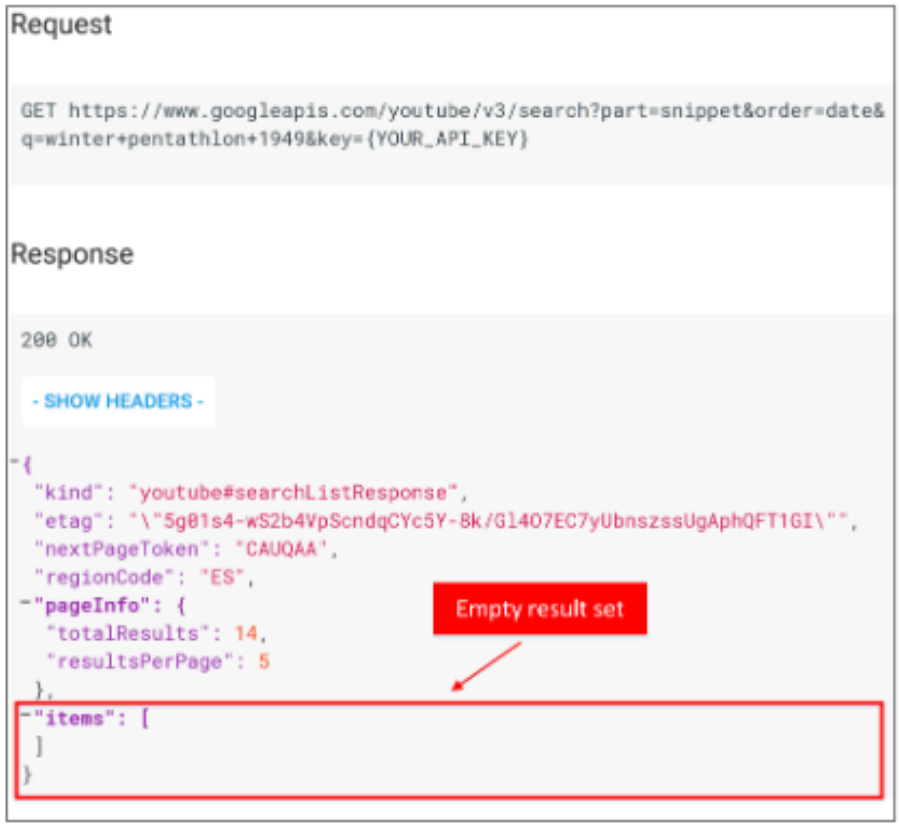
What's more, you know the output for all these cases.
Your tests cover much more of the input space.
Still no accents, though.

What if you didn't have the known output?
Can download any audio from the Internet.
Still have the equality relation with the transformed inputs.

Example: YouTube search



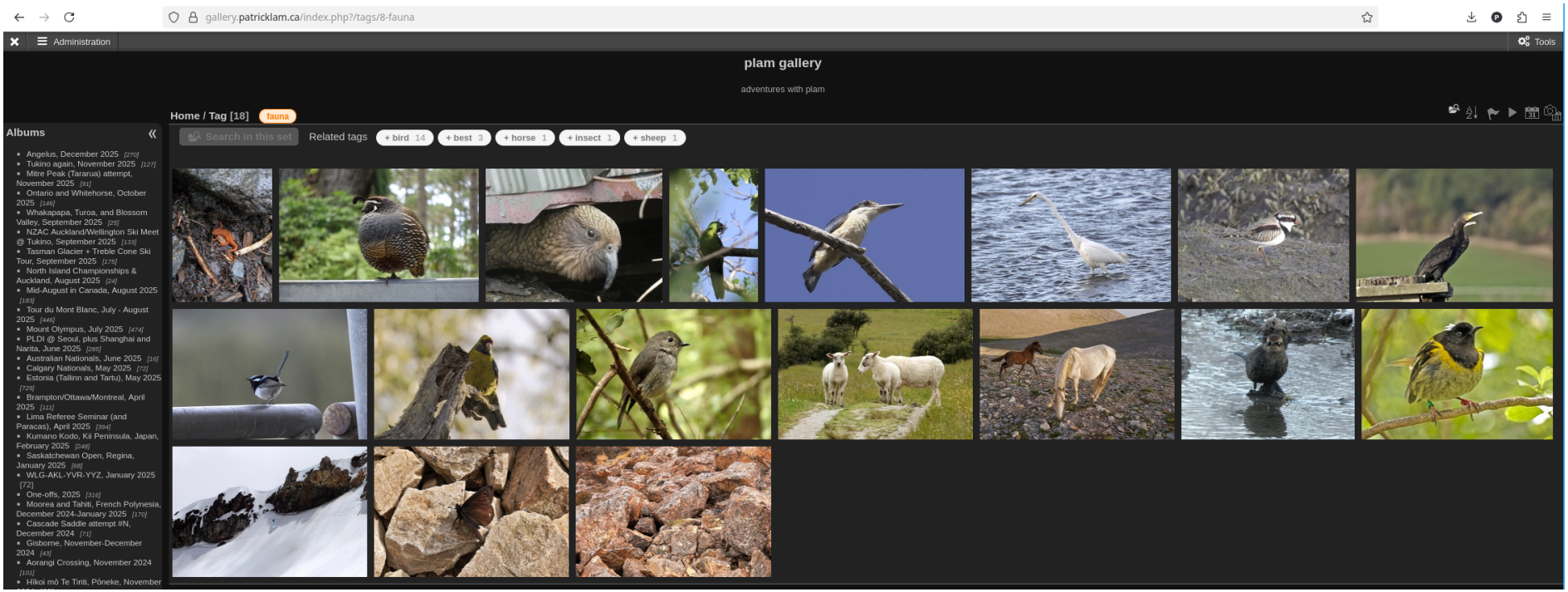
(a) Source test case



(b) Follow-up test case (empty result set)

Figure 4: Metamorphic test revealing a bug in Youtube

Example: tagged image search



Example: tagged image search

For our example: tags “red” and “blue”.
An image may be tagged both “red” and “blue”.

Say there are 4 images.
$\{1, 2, 3\}$ are tagged “red”.
$\{1, 2, 4\}$ are tagged “blue”.

Some tagged image search queries

Query: “has any tag”: $\{$ 1, 2, 3, 4$\}$ (n=4).
Query: “red”: $\{$1, 2, 3$\}$ (n=3).
Query: “blue”: $\{$1, 2, 4$\}$ (n=3).

It must be that

$$3 + 3\ (red + blue) \geq 4\ (any).$$

Metamorphic relation output patterns

A list of patterns from the paper:

- equivalence: same output items, perhaps in different order;
- equality: same output items, same order;
- subset: follow-up output has a subset of original output;
- disjoint: source and follow-up outputs have no items in common;
- complete: union of follow-up outputs completely make up source output;
- difference: source and follow-up outputs differ by a specific set D.

Metamorphic relation output pattern: equivalence

Source input is a query.
Follow-up input is the same query but with some different ordering requested, like “sort by date”.

Expect: same items in outputs, but different order.

Metamorphic relation output pattern: equality

Had this in the `min` example at start; also speech-to-text.

Another example:

Source input is a query.

Follow-up input is the same query but explicitly requesting the default ordering (e.g. sort by relevance).

Expect: same items in outputs, with same order.

Metamorphic relation output pattern: subset

Search engine example with exclusions was like this.

Another example:

Source input is a geolocation query with radius of 50km.
Follow-up input is the same query but a smaller radius.

Expect: follow-up output is subset of source output.

Metamorphic relation output pattern: disjoint

Example:

source input = Spotify albums of “michael buble” from 2012,
follow-up input = Spotify albums of “michael buble” from 2014.

Expect: there should be no items in both the source and the follow-up output sets.

Metamorphic relation output pattern: complete

This is a relation between the source output and the set of follow-up outputs.

Example context: There are short, medium, and long YouTube videos.

If the source input is for keyword “testing”, then one can make three follow-up inputs: short “testing”, medium “testing”, and long “testing”.

Expect: bombined, the results for short, medium, and long “testing” videos should be the same as the results for just “testing”.

Metamorphic relation output pattern: complete

Example:

I upload two videos which I know to be similar except for the length and title. The create operation returns the video uploaded.

Expect: the source and follow-up metadata only differ on length and title, other properties may be the same.

Part II

Code Review