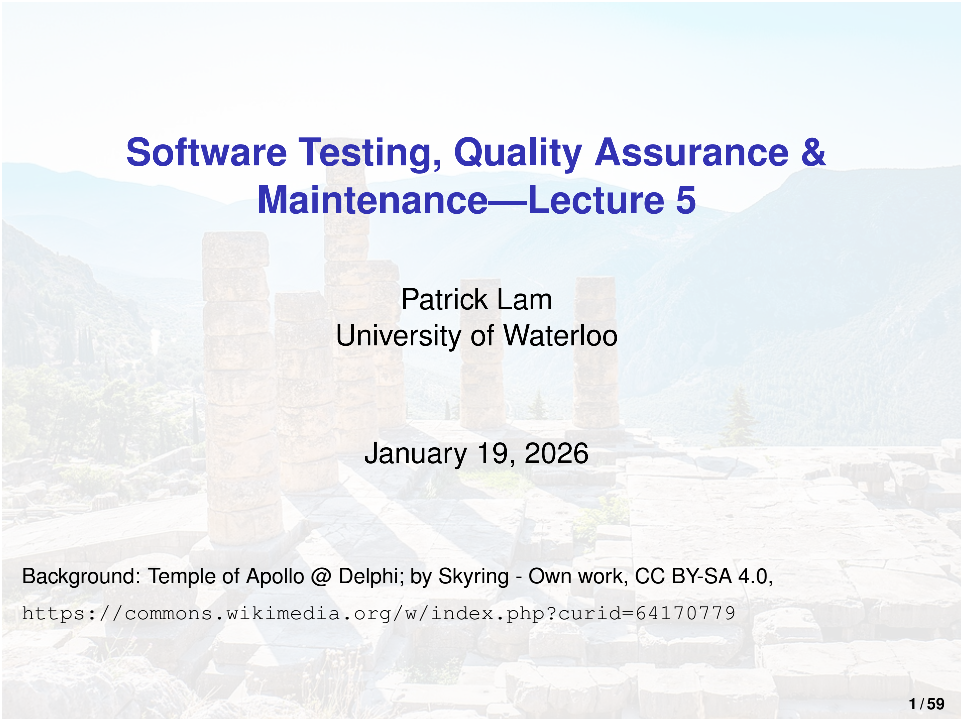


Software Testing, Quality Assurance & Maintenance—Lecture 5

Patrick Lam
University of Waterloo

January 19, 2026

Part I

The Oracle Problem

What’s the right answer?

cop-out: “ask a human”

Begging the Question

Taking the answer the system computes as the right answer.

(is the basis for regression testing, though)

Simple example: add

```python
def add(x, y):
  return x + y
```

We all agree about the output of `add(1,1)`.

Right?

(Almost)

(What about `add("3", 5)` in JavaScript?)

High school math

Consider function `solve_quadratic()` for

$$x^2 - 2x - 4 = 0.$$

Need to read the function name and remember high school math.

Also, edge cases: no solutions;
floating-point shenanigans.

Human Oracles: source of truth

Unit level: Mainly use the function name, plus any function documentation (if it exists).

More generally: You use your human experience to say what the answer should be.

Part II

Helping Human Oracles

Sometimes there is no alternative

You may have to ask a human.

Basis for judgment

Does the output meet the system requirements?

(Eliciting requirements not in scope for this course.)

Easy versus hard inputs

Setting: function `calculate_days_between()`.

What's the answer for:

- 12/24/2025 and 12/25/2025?

What about

- -5455/23195/-30879 and -5460/24100/-30800?

Even if we sanitize/declare negative numbers invalid, some inputs are still easier to check.

Input Profiles

Generate inputs that fit expected input profiles:

Start with developers' sanity-check inputs
(like 12/24/2025 and 12/25/2025 etc).

Also, sanitizing checks inside the code are good places to start.

Months: valid months, 0, -1, 13.

Other options

1. Start from normal inputs,
   use genetic algorithms,
   or generate from distributions.

2. Reuse partial inputs, manually modified;
   change one thing at a time,
   thus, easier to reason about changes in
the output.

   e.g. go from 0/1/2010 to 1/1/2010, etc.

Integers vs strings

Even though some integers aren’t very good (e.g. 23195), it’s still easier to create integers than strings, and easier to create strings than e.g. trees.

Space of strings is bigger;
space of sensible strings is proportionally smaller.

Can use random strings as fuzzed inputs,
but also want strings that pass sanity checks.

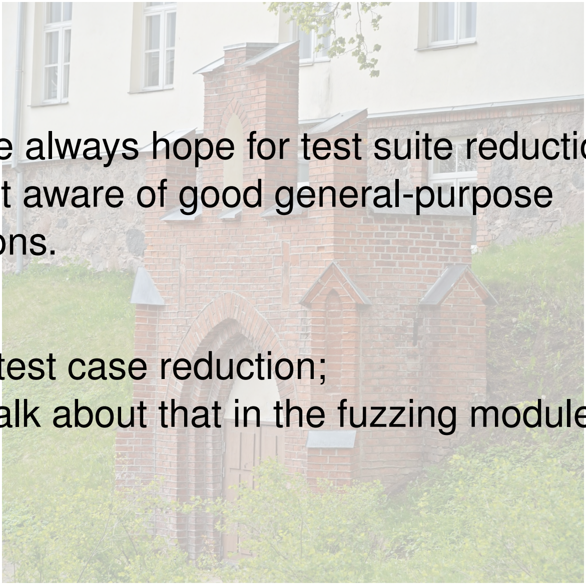
Can mine the web for strings, or generate strings using heuristics (or LLMs).

Crowdsourcing inputs

People have tried to use Mechanical Turk.

Apparently it’s hard to get good results.

Reducing volume of work for human testers



People always hope for test suite reduction.
I’m not aware of good general-purpose solutions.

Also: test case reduction;
we’ll talk about that in the fuzzing module.