

Lecture 12 – Course Review

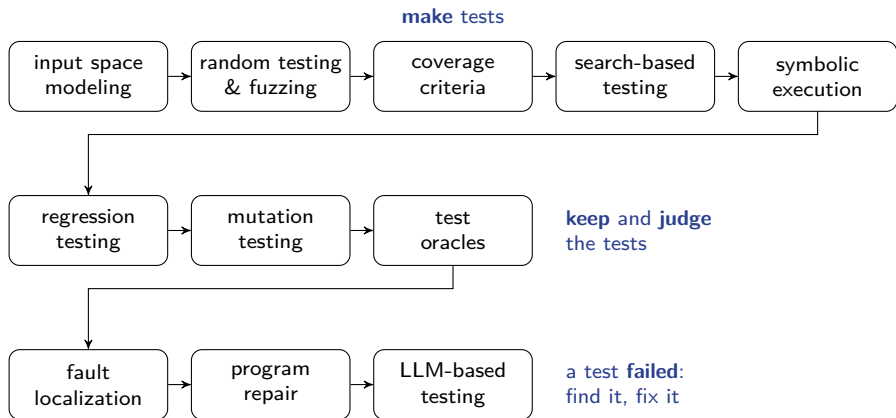
AAA705: Software Testing and Quality Assurance

Jihyeok Park



2026 Fall

Each topic started from the question the previous one left open.



	the generator uses	example
input space	a model of the input	covering array : 3 options, 2 values, 4 rows cover every pair
fuzzing	nothing, then a new-coverage bit	coverage-guided : keep a mutated input if it takes a new branch
search	a distance	branch distance : <code>if (x == 42)</code> , minimize $ x - 42 $
symbolic	a formula	path condition $a \geq 10 \wedge b = 2a \wedge a + b > 100$, SMT solver : (34, 68)

- **Coverage criteria** said what “new” means: node, edge, prime path, clause
- From fuzzing to symbolic execution, the generator learns **more** about the program. All of them answer: **what to feed?**

```
def abs(x):  
    return -x if x < 0 else x  
  
def test(): abs(-3); abs(0); abs(3)
```

- **Regression testing:** `abs(0)` covers nothing new – **minimization**.
After a change, **selection** of the tests it touches, **prioritization** of the rest
- **Mutation testing:** 100% branch coverage, yet it checks nothing. The mutant `-x → x` **survives**; the **mutation score** counts the ones that do not
- **Test oracles:** `assert abs(-1) == 1` – who said 1? Without the answer: a **metamorphic** relation $\sin(x) = \sin(\pi - x)$, a **differential** run of two implementations, or a **property** over random inputs

```
1  int mid(int x, int y, int z) {  
2      int m = z;  
3      if (y < z) {  
4          if (x < y)      m = y;  
5          else if (x < z) m = y; // bug  
6      } else {  
7          if (x > y)      m = y;  
8          else if (x > z) m = x;  
9      }  
10     return m;  
11 }
```

mid(2, 1, 3) returns 1, not 2.

- **Fault localization:** shrink the input (**delta debugging**), rank the lines by passing and failing runs (**SBFL**, Tarantula, Ochiai): line 5 is rank 1
- **Program repair:** generate $m = x$, keep it if all tests pass. Passing all tests = **plausible** $\neq$ **correct**

	the model replaced	it did not replace
inputs	the grammar, the seeds, the mutator	running the program
unit tests	writing the test	build, run, cover, review
oracles	drafting the assertion	checking the assertion
patches	writing the patch	the tests that judge it

- The model is a new **generator**. The **judge** is still the one from this course

question	what we learned
What to feed?	covering arrays, random testing, fuzzing, search-based testing, symbolic execution
Is it enough?	coverage criteria, mutation testing
Which to rerun?	minimization, selection, prioritization
What is right?	metamorphic, differential, property-based testing
What went wrong?	delta debugging, SBFL, program repair

- LLMs made the first row **cheap**. The other four are how we **judge** what they generate – and that is what you learned here

- I hope you enjoyed the class!

Jihyeok Park

jihyeok_park@korea.ac.kr

<https://plrg.korea.ac.kr>