LASSO – Automated Mass-Analysis of Code and Tests

UNIVERSITÄT MANNHEIM

Problem: GCD (Greatest Common Divisor)
Interface: *gcd(int, int)->int*

`test(p1:Class, p2:int, p3:int)`

	A	B	C	D	E
1		create	?p1		
2		gcd	A1	?p2	?p3

output *operation* *inputs*

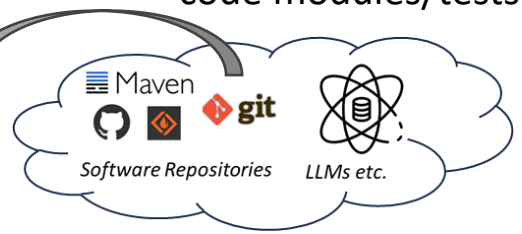
Sequence Sheet

Tests

N versions (candidate implementations)

	V1	V2	V3	...	V_n
test1	*test(V1,03,07)*	*test(V2, 03,07)*	*test(V3, 03,07)*		*test(V_n, 03,07)*
test2	*test(V1,10,15)*	*test(V2,10,15)*	*test(V3,10,15)*		*test(V_n,10,15)*
test3	*test(V1,49,14)*	*test(V2,49,14)*	*test(V3,49,14)*		*test(V_n,49,14)*
...					
$test_m$	*$test_m$(V1, x, y)*	*$test_m$(V2, x, y)*	*$test_m$(V3, x, y)*		*$test_m$(V_n, x, y)*

code modules/tests



Stimulus Matrix (SM)

Pipeline Script

Test Driver "Arena"

	A	B	C	D	E
1		create	**V1**		
2	1	gcd	A1	3	7

	V1		V2		V3		...	V_n	
test1	*test(V1,03,07)*	*1*	*test(V2,03,07)*	*7*	*test(V3,03,07)*	*-7*		*test(V_n,03,07)*	...
test2	*test(V1,10,15)*	*5*	*test(V2,10,15)*	*15*	*test(V3,10,15)*	*-15*		*test(V_n,10,15)*	...
test3	*test(V1,49,14)*	*7*	*test(V2,49,14)*	*14*	*test(V3,49,14)*	*28*		*test(V_n,49,14)*	...
...									
$test_m$	*$test_m$(V1, x, y)*	...	*$test_m$(V2, x, y)*	...	*$test_m$(V3, x, y)*	...		*$test_m$(V_n, x, y)*	...

	Time	Memory	Method Calls	Branches Covered	...
	...	...	...	...	...

Large-Scale Software Observatorium (LASSO Platform)

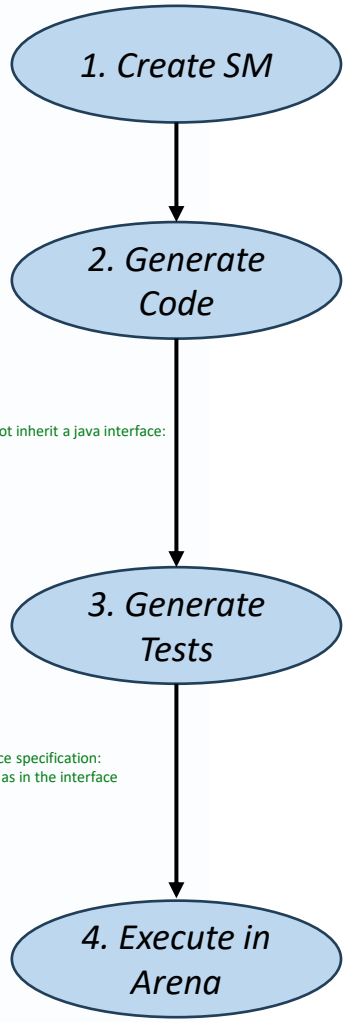
Stimulus Response Matrix (SRM)

Differential Testing (identify discrepancies)

LSL Pipelines & Stimulus Response Matrices

UNIVERSITÄT MANNHEIM

```
dataSource 'lasso_quickstart'
study(name: 'OpenAI-DGAI') {

  profile('java17Profile') {
    scope('class') { type = 'class' }
    environment('java17') {
      image = 'maven:3.9-eclipse-temurin-17'
    }
  }

  def humanEval = loadBenchmark("humaneval-java-reworded")

  action(name: "createStimulusMatrices") {
    execute {
      def myProblems = [humanEval.abstractions['HumanEval_13_greatest_common_divisor']]
      myProblems.each { problem ->
        stimulusMatrix(problem.id, problem.lql, [/*impls*/], problem.tests) // id, interface, impls, tests
      }
    }
  }

  action(name: 'generateCodeGpt', type: 'GenerateCodeOpenAI') {
    dependsOn 'createStimulusMatrices'
    include '*'
    profile('java17Profile')

    apiKey = "demo"
    model = "gpt-4o-mini"
    samples = 1

    prompt { stimulusMatrix ->
      def prompt = [:] // create prompt model
      prompt.promptContent = """implement a java class with the following interface specification, but do not inherit a java interface:
```${stimulusMatrix.lql}```. Only output the java class and nothing else."""
      return [prompt] // list of prompts
    }
  }

  action(name: 'generateTestsGpt', type: 'GenerateTestsOpenAI') {
    dependsOn 'generateCodeGpt'
    include '*'
    profile('java17Profile')

    apiKey = "demo"
    model = "gpt-4o-mini"
    samples = 1

    prompt { stimulusMatrix ->
      def prompt = [:] // create prompt model
      prompt.promptContent = """generate a junit test class to test the functionality of the following interface specification:
```${stimulusMatrix.lql}```. Assume that the specification is encapsulated in a class that uses the same naming as in the interface
specification. Only output the JUnit test class and nothing else."""
      return [prompt] // list of prompts
    }
  }

  action(name: 'execute', type: 'Arena') {
    features = ["cc"] // coverage: code coverage, mutation testing etc.

    dependsOn 'generateTestsGpt'
    include '*'
    profile('java17Profile')
  }
}
```



test1
test2
test3

New **stimulus matrix** from coding problem

	V1	V2	…	V_n
test1				
test2				
test3				

Add **implementations** (versions)

	V1	V2	…	V_n
test1				
test2				
…				
$test_m$				

Add **tests**

	V1		V2		…	V_n	
test1	*test(V1,03,07)*	*1*	*test(V2,03,07)*	*7*		*test(V_n,03,07)*	…
test2	*test(V1,10,15)*	*5*	*test(V2,10,15)*	*15*		*test(V_n,10,15)*	…
…							
$test_m$	*$test_m$(V1, x, y)*	…	*$test_m$(V2, x, y)*	…		*$test_m$(V_n, x, y)*	…

Observe **outputs, metrics** …