# Fault Models

## Halstead

### Difficulty

* 1. Operands are missing from the Token list and are unaccounted for in the tests
  2. Difficulty is not calculated due to the helper log function.
     1. Abandoning that helper function and doing calculations in the finishTree()

### Length

* 1. Operands are missing from the Token list and are unaccounted for in the tests

### Effort

* 1. Operands are missing from the Token list and are unaccounted for in the tests

3.2 Effort is not calculated due to the helper log function.

* + 1. Abandoning that helper function and doing calculations in the finishTree()

3.3 Not counting the unique tokens, operands, and operators

* + 1. Utilizing a hash set to work around this issue.

### Vocabulary

* 1. Valid operands are unaccounted for even though they are in the Token set
     1. Removing the helper `contains ()` function and utilizing the hash set

### Volume

5.1Operands are missing from the Token list and are unaccounted for in the tests

* 1. Volume is not calculated correctly
     1. Removing the helper log function

## Number of Checks

### Comments

* 1. Number of comments are calculated incorrectly
     1. Changing the token type to COMMENT\_CONTENT rather than single line and begin comment block

### Expression

* 1. Expression is not calculated correctly
     1. Removing count = 0 in FinishTree

### Line Comment

### Loop

* 1. Having a hard time retrieving accepted, required tokens
     1. Missing a for\_loop token in the types. Adding that fixed the issue.

### Operand

* 1. Operands are missing
     1. Adding and updating operands in the token types along with related files

### Operator

* 1. Operators are missing
     1. Adding and updating operators in the token types along with related files

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PIT Results |  |  |  |  |
|  |  |  |  |  |
| Number of Classes | Line Coverage | Mutation Coverage | Test Strength |  |
| 11 | 100% | 72% | 72% |  |
|  | 424/426 | 63/87 | 63/87 |  |
|  |  |  |  |  |
| Breakdown by Package | |  |  |  |
|  |  |  |  |  |
| Name | Number of Classes | Line Coverage | Mutation Coverage | Test Strength |
| HalsteadChecks | 5 | 99% | 70% | 70% |
|  |  | 319/321 | 37/53 | 37/53 |
| NumOfChecks | 6 | 100% | 76% | 76% |
|  |  | 105/105 | 26/34 | 26/34 |

A picture containing timeline

Description automatically generated

===========================================================================

- Statistics

===========================================================================

>> Line Coverage: 424/426 (100%)

>> Generated 87 mutations Killed 63 (72%)

>> Mutations with no coverage 0. Test strength 72%

>> Ran 153 tests (1.76 tests per mutation)

# Black Box Testing

I’ve thought covering lines would cover most, if not all logic but after writing black box tests I was proven wrong. I realized I didn’t count unique tokens for some of the checks, I was counting unary operators wrong, I didn’t include array operations.

Black box testing helped me find holes in my logic, pointed out things that were obvious I didn’t think of checking. Like missing operands, operators, me randomly setting count back to zero and spending hours trying to figure out why it is not returning the correct amount.

I included the test drivers in each black box rather than having a test driver as a package, hindsight, that would’ve been a better approach to this.