Introduction
Problem
Pseudo-tested Methods
Function-Fiasco
Evaluation Strategy
Results
Conclusion

Automatic Detection of Pseudo-Tested Methods in a Test Suite Using Fault Injection

Nicholas Tocci

April 3, 2019

Problem

Introduction
Problem
Coverage
Pseudo-tested Methods
Function-Fiasco
Evaluation Strategy
Results
Conclusion

How can we know if our test suites are effective?

Coverage

Coverage

Def: % of a system that has been tested.

Calculation

Introduction
Problem
Coverage
Calculation
Coverage vs Adequate Coverage
Pseudo-tested Methods
Function-Fiasco
Evaluation Strategy
Results
Conclusion

$$FunctionCoverage = \frac{NumberofTestedMethods}{TotalNumberofMethods}$$

High Coverage

Introduction
Problem
Coverage Calculation
Coverage vs Adequate Coverage
Pseudo-tested Methods
Function-Fiasco
Evaluation Strategy
Results
Conclusion

Introduction
Problem
Pseudo-tested Methods
Defintion
Detection
Function-Fiasco
Evaluation Strategy
Results
Conclusion

Pseudo-tested Methods

Definition

Introduction
Problem
Pseudo-tested Methods
Defintion
Detection
Function-Fiasco
Evaluation Strategy
Results
Conclusion

What is a Pseudo-tested Method?

Def: It will never fail.

Detection

Introduction
Problem
Pseudo-tested Methods
Defintion
Detection
Function-Fiasco
Evaluation Strategy
Results
Conclusion

How Can We Detect Pseudo-tested Methods

It is harder than you think!

Example of a Pseudo-tested method

Introduction
Problem
Pseudo-tested Methods
Defintion
Detection
Function-Fiasco
Evaluation Strategy
Results
Conclusion

```
numbers.py:
def numberOrder(n):
    numbersSorted = sorted(n)
    return numbersSorted


test_numbers.py:
def test_numbers_ordered():
    numbers = set([2,4,3,1])
    sortedNumbers = set([1,2,3,4])
    orderedNumbers = numberOrder(numbers)
    assert numbers == sortedNumbers
```

What is Function-Fiasco

A Pseudo-tested method detection tool

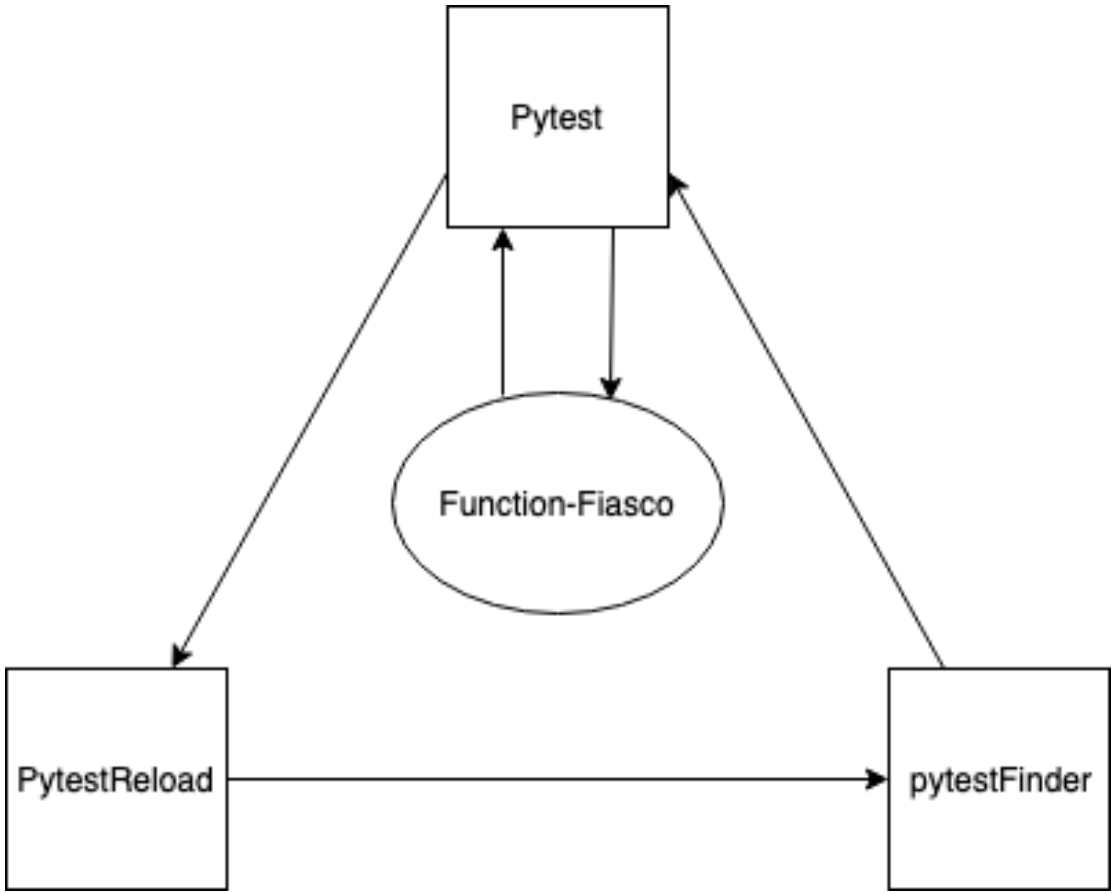
Decorator Function

Introduction
Problem
Pseudo-tested Methods
Function-Fiasco
What is Function-Fiasco
Flow
Evaluation Strategy
Results
Conclusion

```
def skipper(func):
    functionsComplete = globs.functionsComplete
    checked = checkFunctionsComplete(func, functionsComplete)
    globs.checked = str(checked)
    if checked == True and globs.firstExe == True:
        def wrapper(*args, **kwargs):
            checkType(var, func.__name__)
            return var
        return wrapper
    elif checked == True and globs.firstExe == False:
        def doFunc(*args, **kwargs):
            var = func(*args, **kwargs)
            return checkType(var, func.__name__)
        return doFunc
    else:
        def doFunc(*args, **kwargs):
            var = func(*args, **kwargs)
            return var
        return doFunc
```

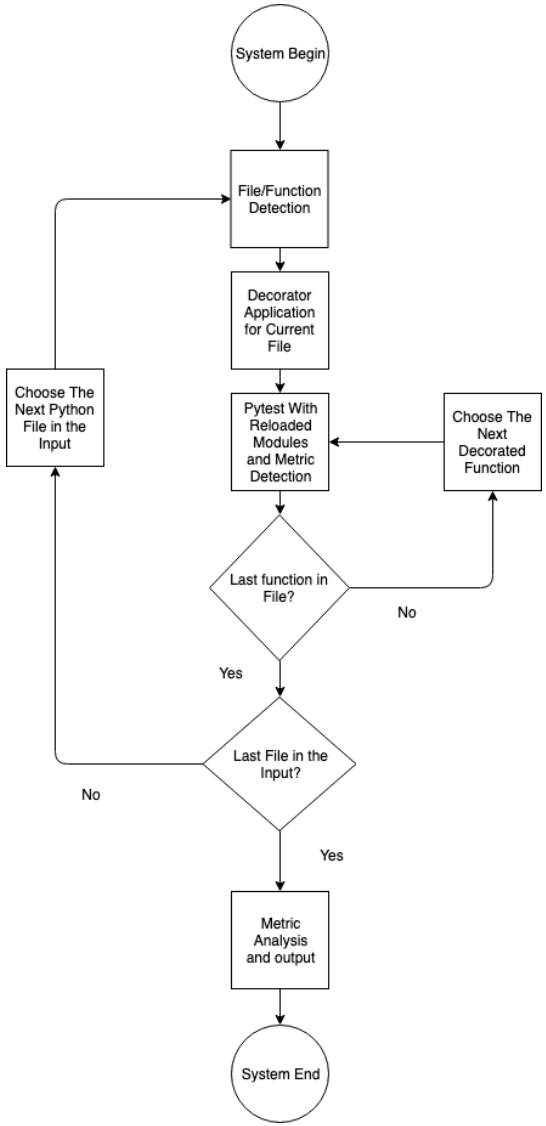
Execution Flow

Introduction
Problem
Pseudo-tested Methods
Function-Fiasco
What is Function-Fiasco
Flow
Evaluation Strategy
Results
Conclusion



Flow to system

Introduction
Problem
Pseudo-tested Methods
Function-Fiasco
What is Function-Fiasco
Flow
Evaluation Strategy
Results
Conclusion



Coverage Calculation

Introduction
Problem
Pseudo-tested Methods
Function-Fiasco
Evaluation Strategy
Coverage Calculation
Truly-Tested-Method Calculation
Metrics Produced
Results
Conclusion

$$FunctionCoverage = \frac{NumberofTestedMethods}{TotalNumberofMethods}$$

Coverage Example

Introduction
Problem
Pseudo-tested Methods
Function-Fiasco
Evaluation Strategy
Coverage Calculation
Truly-Tested-Method Calculation
Metrics Produced
Results
Conclusion

NUMM	NUMTM	Function Coverage
40	25	62.5%

Truly-Tested-Method Calculation

Introduction
Problem
Pseudo-tested Methods
Function-Fiasco
Evaluation Strategy
Coverage Calculation
Truly-Tested-Method Calculation
Metrics Produced
Results
Conclusion

- **Number of Truly-Tested-Methods = NUMTTM**
- **Number of Tested Methods = NUMTM**
- **Number of Pseudo-tested Methods = NUMPTM**

$$NUMTTM = NUMTM - NUMPTM$$

Truly-Tested-Method Example

Introduction
Problem
Pseudo-tested Methods
Function-Fiasco
Evaluation Strategy
Coverage Calculation
Truly-Tested-Method Calculation
Metrics Produced
Results
Conclusion

NUMTM	NUMPTM	NUMTTM
25	3	22

Updated-Coverage Calculation

Introduction
Problem
Pseudo-tested Methods
Function-Fiasco
Evaluation Strategy
Coverage Calculation
Truly-Tested-Method Calculation
Metrics Produced
Results
Conclusion

$$UC = \frac{NumberofTrulyTestedMethods}{TotalNumberofMethods}$$

Output

Introduction

Problem

Pseudo-tested Methods

Function-Fiasco

Evaluation Strategy

Coverage Calculation

Truly-Tested-Method Calculation

Metrics Produced

Results

Conclusion

Statement Coverage	Initial Function coverage	Number of Methods	Number of Tested Methods	Fiascoed Methods	Number of Pseudo−tested Methods	Number of Truly Tested Methods	Updated Coverage
67%	0.44	730	319	16	9	310	0.42

List of Systems

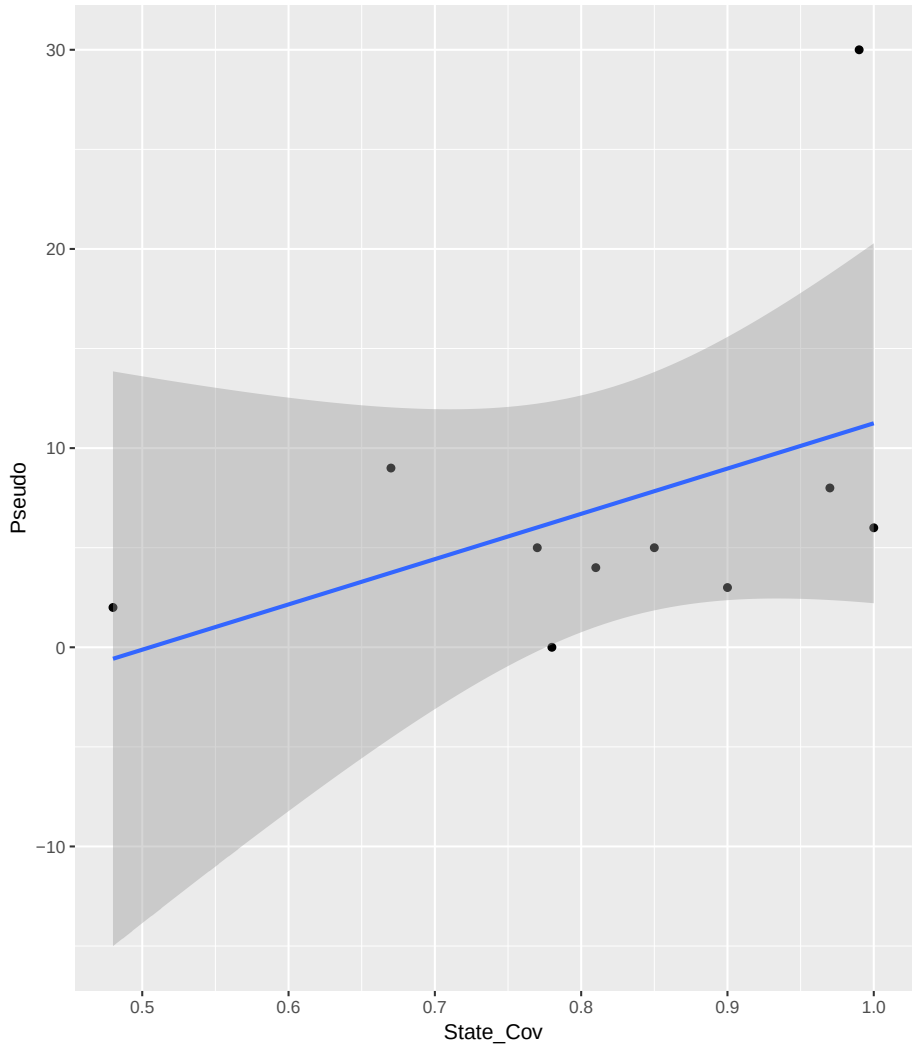
Introduction
Problem
Pseudo-tested Methods
Function-Fiasco
Evaluation Strategy
Results
Conclusion

	System_Name	Num_Methods	Fiascoed	Num_Tests
1	Hashids-Python	16	10	59
2	Bleach	368	8	312
3	Pycco	22	6	17
4	Howdoi	20	2	18
5	Flashtext	42	7	23
6	Honcho	58	7	124
7	Maya	88	13	277
8	Gator	91	53	505
9	Hatch	134	14	339
10	Nikola	732	16	205

Table: List of systems used for testing.

Statement Coverage

Introduction
Problem
Pseudo-tested Methods
Function-Fiasco
Evaluation Strategy
Results
Conclusion



Type Risk

Introduction
Problem
Pseudo-tested Methods
Function-Fiasco
Evaluation Strategy
Results
Conclusion

	Type	Used	Found	ratio
1	Strings	69	33	47.8%
2	Booleans	49	35	73.5%
3	Ints	18	4	22.2%
4	Floats	1	0	0%

Table: Breakdown of the number of pseudo-tested method per type.

Field Connects

Introduction
Problem
Pseudo-tested Methods
Function-Fiasco
Evaluation Strategy
Results
Conclusion
Impact
Future Research
Demo

- Computer Science

Field Connects

Introduction
Problem
Pseudo-tested Methods
Function-Fiasco
Evaluation Strategy
Results
Conclusion
Impact
Future Research
Demo

- Computer Science
- Software Engineering

Field Connects

Introduction
Problem
Pseudo-tested Methods
Function-Fiasco
Evaluation Strategy
Results
Conclusion
Impact
Future Research
Demo

- Computer Science
- Software Engineering
- Software Testing

What is the impact of this research?

Introduction
Problem
Pseudo-tested Methods
Function-Fiasco
Evaluation Strategy
Results
Conclusion
Impact
Future Research
Demo

Coverage with fault detection

Better Understanding of Pseudo-tested Methods

Automatic Detection Tool

Future Research

Introduction
Problem
Pseudo-tested Methods
Function-Fiasco
Evaluation Strategy
Results
Conclusion
Impact
Future Research
Demo

- Different Types and Paramaterization
- Bug Fixes
- Documentation
- Further Testing

Introduction
Problem
Pseudo-tested Methods
Function-Fiasco
Evaluation Strategy
Results
Conclusion
Impact
Future Research
Demo

Function-Fiasco

Automatic Detection System for Pseudo-tested Methods