Inline Tests

Yuki Liu, Pengyu Nie, Owolabi Legunsen, Milos Gligoric

October 12, 2022
ASE, Michigan, USA

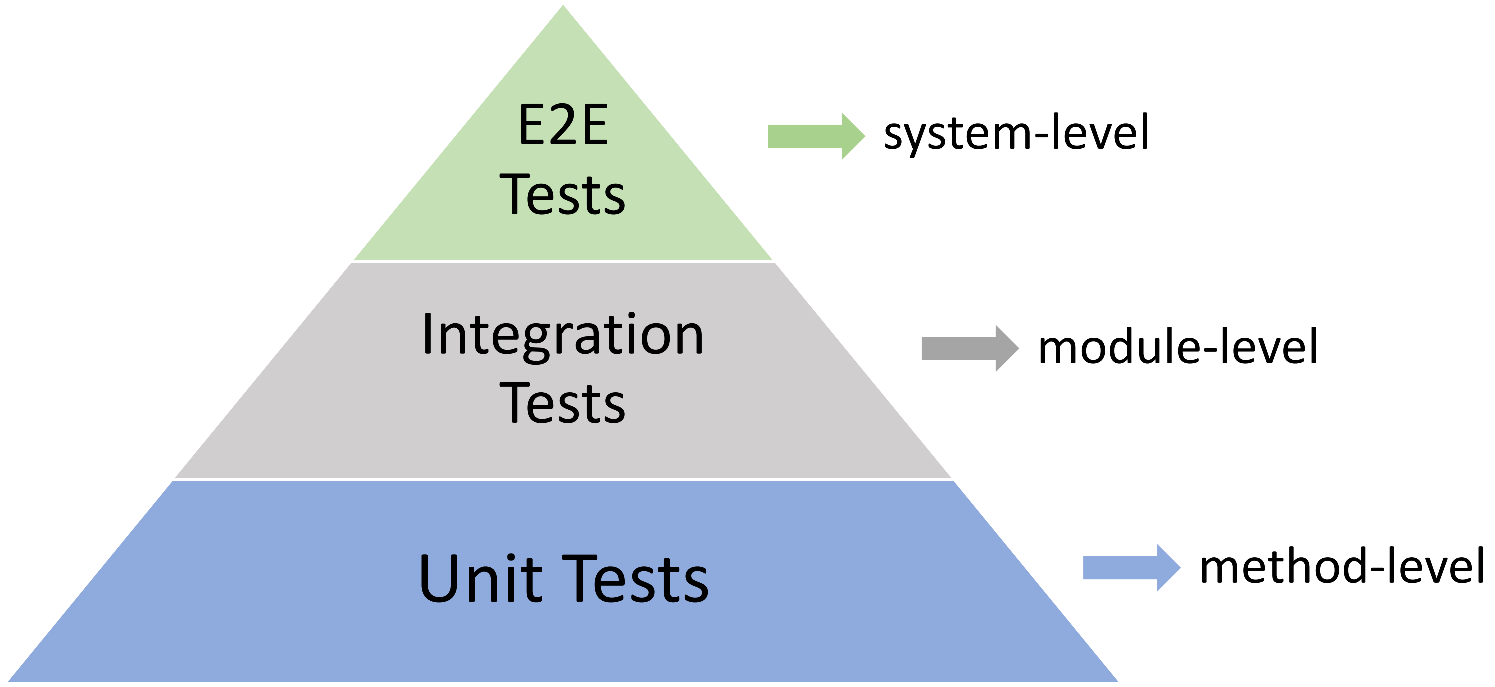
TEXAS
The University of Texas at Austin

Cornell University

NSF
652517, 2019277,
2045596, 2107291,
2217696

Current Levels of Test Granularity

What if we want to test a single statement in a method?



Value of Testing Individual Statements

https://github.com/noDRM/DeDRM_tools/blob/master/DeDRM_plugin/k4mobidedrm.py

```python
def decryptBook(infile, outdir, kDatabaseFiles, androidFiles, serials, pids):
...
    return 0
```

Value of Testing Individual Statements

https://github.com/noDRM/DeDRM_tools/blob/master/DeDRM_plugin/k4mobidedrm.py

```
def decryptBook(infile, outdir, kDatabaseFiles, androidFiles, serials, pids):
...
    # Try to infer a reasonable name
    orig_fn_root = os.path.splitext(os.path.basename(infile))[0]
    if (    ✗ {0-9A-F}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}   ✓ ec69ba8e-0bfe-4f4b-a8cf-bfc313a97e55
        re.match('^B[A-Z0-9]{9}(_EBOK|_EBSP|_sample)?$', orig_fn_root) or
        re.match('^{0-9A-F-}{36}$', orig_fn_root)
    ):  # Kindle for PC / Mac / Android / Fire / iOS
        clean_title = cleanup_name(book.getBookTitle())
        outfilename = "{}_{}".format(orig_fn_root, clean_title)
...
    return 0
```

bug

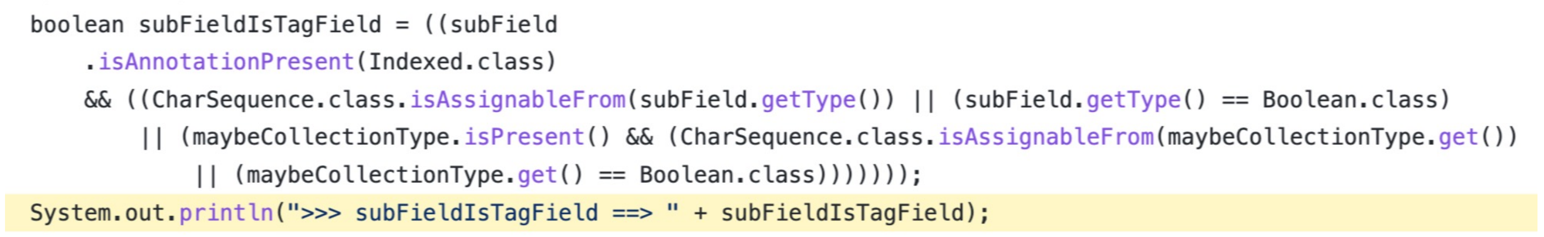
Developers Want to Test Code Within Methods

- Single-statement bugs occur frequently [1, 2]
  - Hard-to-understand
  - Complex program logic
- Unit tests rarely fail single-statement bugs [3]
  - Statements buried deeply inside complicated program logic

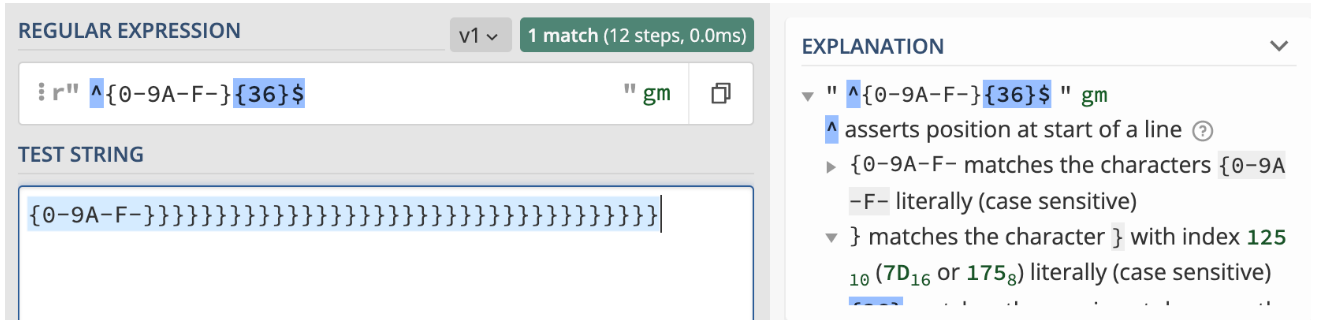
[1] Arthur V Kamienski, Luisa Palechor, Cor-Paul Bezemer, and Abram Hindle. 2021. PySStuBs: Characterizing single-statement bugs in popular open-source Python projects. In MSR. 520–524.
[2] Rafael-Michael Karampatsis and Charles Sutton. 2020. How often do single-statement bugs occur? The ManySStuBs4J dataset. In MSR. 573–577.
[3] Jasmine Latendresse, Rabe Abdalkareem, Diego Elias Costa, and Emad Shihab. 2021. How effective is continuous integration in indicating single-statement bugs?. In MSR. 500–504.

Existing Approaches

- Developers use printf debugging, website, in-IDE popups, etc.



```java
boolean subFieldIsTagField = ((subField
    .isAnnotationPresent(Indexed.class)
    && ((CharSequence.class.isAssignableFrom(subField.getType()) || (subField.getType() == Boolean.class)
        || (maybeCollectionType.isPresent() && (CharSequence.class.isAssignableFrom(maybeCollectionType.get())
            || (maybeCollectionType.get() == Boolean.class)))))));
System.out.println(">>> subFieldIsTagField ==> " + subFieldIsTagField);
```



REGULAR EXPRESSION v1 1 match (12 steps, 0.0ms)

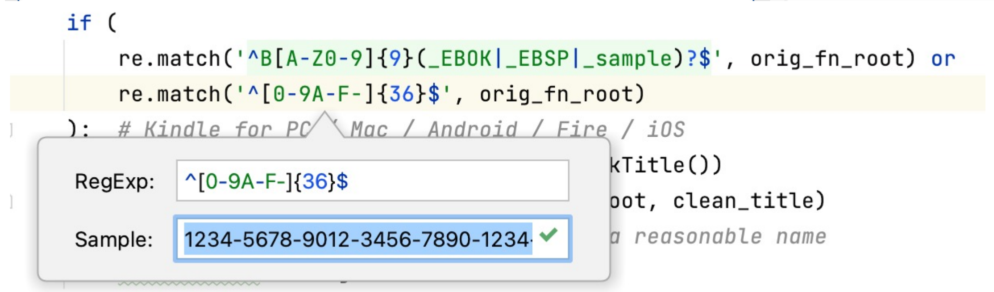
```
r" ^{0-9A-F-}{36}$ " gm
```

TEST STRING

```
{0-9A-F-}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}
```

EXPLANATION

" ^{0-9A-F-}{36}$ " gm
^ asserts position at start of a line
{0-9A-F- matches the characters {0-9A-F- literally (case sensitive)
} matches the character } with index 125₁₀ (7D₁₆ or 175₈) literally (case sensitive)

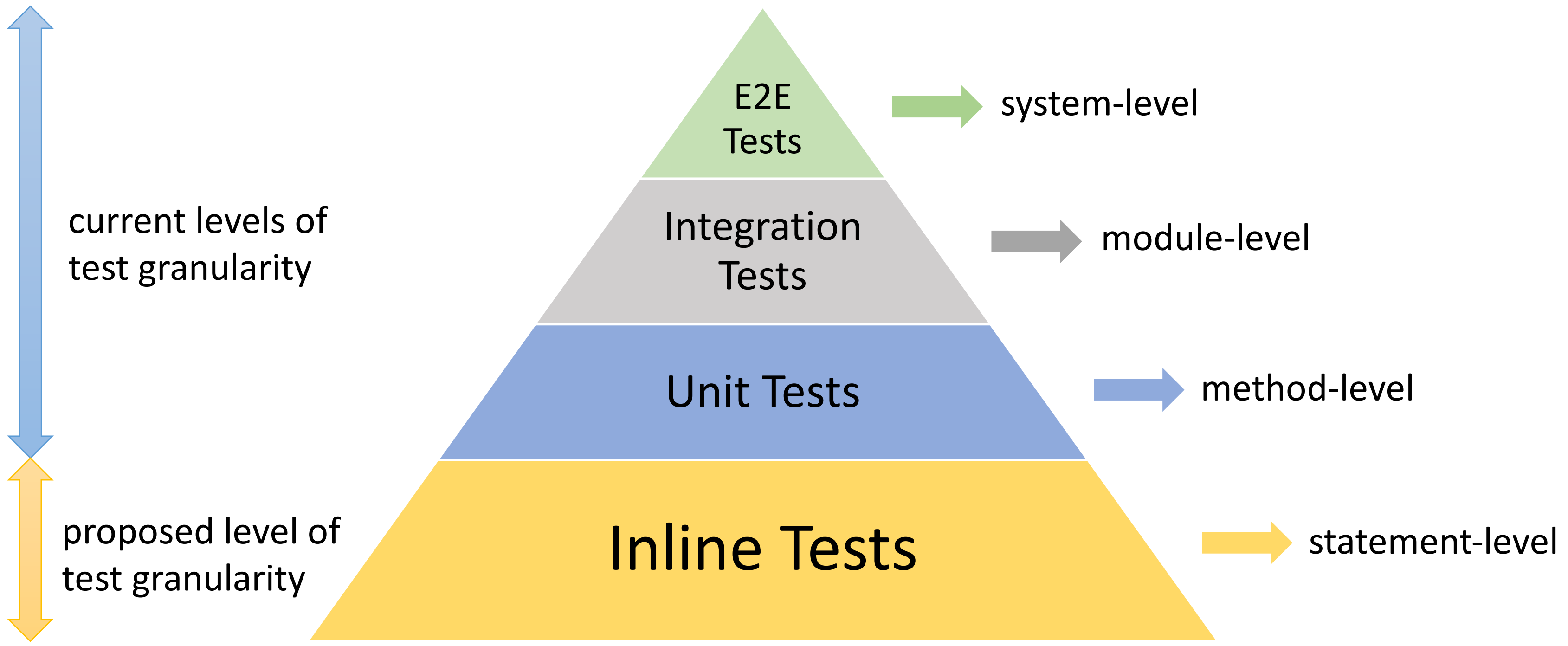


```python
if (
    re.match('^B[A-Z0-9]{9}(_EBOK|_EBSP|_sample)?$', orig_fn_root) or
    re.match('^[0-9A-F-]{36}$', orig_fn_root)
):  # Kindle for PC / Mac / Android / Fire / iOS
                                           kTitle())
                                           oot, clean_title)
                                           a reasonable name
```

RegExp: ^[0-9A-F-]{36}$
Sample: 1234-5678-9012-3456-7890-1234 ✓

- Copy source code
- Leave development environment
- Cannot store results

Inline Tests



Our Contributions

- Idea: introduce a new type of tests, inline tests
- Framework: implement I-Test, the first inline testing framework
- Performance evaluation: measure runtime costs of I-Test
- User study: evaluate programmer perceptions about inline testing

Inline Test Example

https://github.com/noDRM/DeDRM_tools/blob/master/DeDRM_plugin/k4mobidedrm.py

```
def decryptBook(infile, outdir, kDatabaseFiles, androidFiles, serials, pids):
...
    if (
        re.match('^B[A-Z0-9]{9}(_EBOK|_EBSP|_sample)?$', orig_fn_root) or
        re.match('^{0-9A-F-}{36}$', orig_fn_root)
            ):  # Kindle for PC / Mac / Android / Fire / iOS
Declare Here().given(orig_fn_root, 'ec69ba8e-0bfe-4f4b-a8cf-bfc313a97e55').check_true(Group(1))
        clean_title = cleanup_name(book.getBookTitle())          Assign          Assert
        outfilename = "{}_{}".format(orig_fn_root, clean_title)
...
    return 0
```

Inline Test Example

https://github.com/noDRM/DeDRM_tools/blob/master/DeDRM_plugin/k4mobidedrm.py

```
def decryptBook(infile, outdir, kDatabaseFiles, androidFiles, serials, pids):
...
    if (
        re.match('^B[A-Z0-9]{9}(_EBOK|_EBSP|_sample)?$', orig_fn_root) or
-       re.match('^{0-9A-F-}{36}$', orig_fn_root)
+       re.match('^[0-9A-F-]{36}$', orig_fn_root)
                ):  # Kindle for PC / Mac / Android / Fire / iOS
        Here().given(orig_fn_root, 'ec69ba8e-0bfe-4f4b-a8cf-bfc313a97e55').check_true(Group(1))
        clean_title = cleanup_name(book.getBookTitle())
        outfilename = "{}_{}".format(orig_fn_root, clean_title)
...
    return 0
```

I-Test API (Subset)

- Declaration
  - Here()
- Assignment
  - given(variable, value)
- Assertion
  - check_eq(actual, expected)
  - check_true(actual)
  - check_false(actual)

Design of I-Test

- Write code instead of comments
- Write inline tests below the target statement instead of a separate file
- Check only one target statement instead of multiple statements
- Enable during testing and disable in production
- More requirements see paper Section 3

I-Test Implementation

k4mobidedrm.py

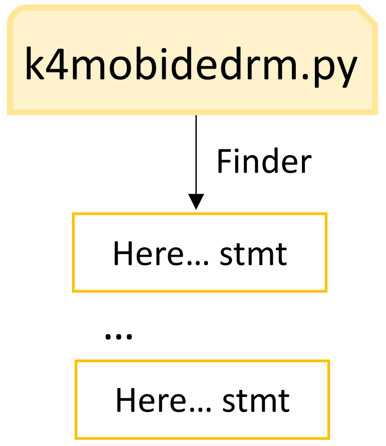
- Given a source file

```
 from inline import Here
...
 def decryptBook(infile, outdir, kDatabaseFiles, androidFiles, serials, pids):
...
     if (
         re.match('^B[A-Z0-9]{9}(_EBOK|_EBSP|_sample)?$', orig_fn_root) or
         re.match('^{0-9A-F-}{36}$', orig_fn_root)
                ):  # Kindle for PC / Mac / Android / Fire / iOS
         Here().given(orig_fn_root, 'ec69ba8e-0bfe-4f4b-a8cf-bfc313a97e55').check_true(Group(1))
         clean_title = cleanup_name(book.getBookTitle())
         outfilename = "{}_{}".format(orig_fn_root, clean_title)
...
     return 0
```

I-Test Implementation

- Finder searches for
  - import statement of Here
  - statements that start with Here

```
 from inline import Here
...
 def decryptBook(infile, outdir, kDatabaseFiles, androidFiles, serials, pids):
...
     if (
         re.match('^B[A-Z0-9]{9}(_EBOK|_EBSP|_sample)?$', orig_fn_root) or
         re.match('^{0-9A-F-}{36}$', orig_fn_root)
                ):  # Kindle for PC / Mac / Android / Fire / iOS
         Here().given(orig_fn_root, 'ec69ba8e-0bfe-4f4b-a8cf-bfc313a97e55').check_true(Group(1))
         clean_title = cleanup_name(book.getBookTitle())
         outfilename = "{}_{}".format(orig_fn_root, clean_title)
...
     return 0
```



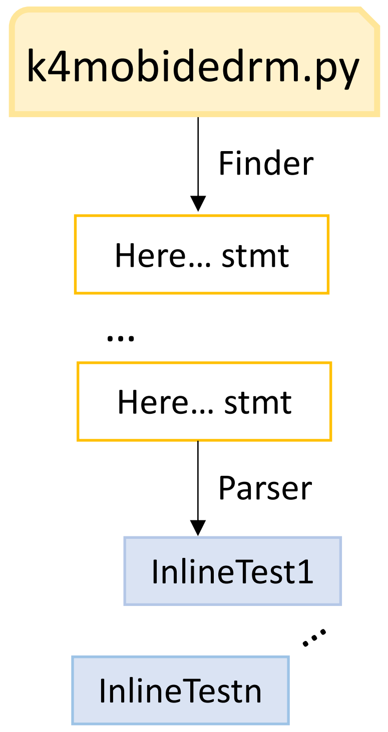
I-Test Implementation

- Parser parses inline test statement to an executable test

```
if (
    re.match('^B[A-Z0-9]{9}(_EBOK|_EBSP|_sample)?$', orig_fn_root) or
    re.match('^{0-9A-F-}{36}$', orig_fn_root)
):  # Kindle for PC / Mac / Android / Fire / iOS
    Here().given(orig_fn_root, 'ec69ba8e-0bfe-4f4b-a8cf-bfc313a97e55')
              .check_true(Group(1))
```

```
orig_fn_root = 'ec69ba8e-0bfe-4f4b-a8cf-bfc313a97e55'
assert re.match('^{0-9A-F-}{36}$', orig_fn_root) == True
```

parsed test



I-Test Implementation

- Runner executes the parsed test

```
orig_fn_root = 'ec69ba8e-0bfe-4f4b-a8cf-bfc313a97e55'
assert re.match('^{0-9A-F-}{36}$', orig_fn_root) == True
```

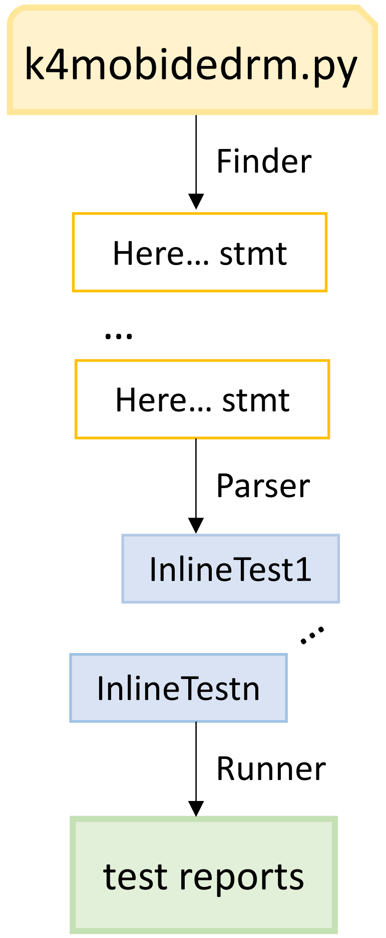
parsed test

compile and execute

```
tree = ast.parse(test.str())
codeobj = compile(tree, filename="<ast>", mode="exec")
exec(codeobj, test.globs)
```

integrate with pytest

```
=========================== test session starts ===========================
platform linux -- Python 3.9.13, pytest-7.1.3, pluggy-1.0.0
collected 1 item
k4mobidedrm.py .                                                    [100%]
=========================== 1 passed in 0.02s ============================
```



Sets of Target Statements

- Regular expression
- String manipulation
- Bit manipulation
- Collection handling (Python only)
- Stream (Java only)

Sets of Target Statements

- Regular expression
  - Python re package, Java java.util.regex package
- String manipulation
- Bit manipulation
- Collection handling (Python only)
- Stream (Java only)

Sets of Target Statements

- Regular expression
- String manipulation
  - string concatenation, string split, string formatting, etc.
- Bit manipulation
- Collection handling (Python only)
- Stream (Java only)

Sets of Target Statements

- Regular expression
- String manipulation
- Bit manipulation
  - left shift(<<), right shift(>>), bitwise and(&), bitwise or(|), bitwise not(~), bitwise XOR(^)
- Collection handling (Python only)
- Stream (Java only)

Sets of Target Statements

- Regular expression
- String manipulation
- Bit manipulation
- Collection handling (Python only)
  - list, set, dict, tuple, etc.
- Stream (Java only)

Sets of Target Statements

- Regular expression
- String manipulation
- Bit manipulation
- Collection handling (Python only)
- Stream (Java only)
  - stream(), filter(), collect(), count(), findFirst(), etc.

Evaluation Setup

- Search 100 top-starred Python and Java projects on GitHub
- Write 87 Python and 65 Java inline tests

Breakdown of 50 Python Examples

Type	# Projects	# Target Statements	# Inline Tests
Regex	15	19	22
String	13	30	32
Bit	15	26	27
Collection	4	5	6
Total	31	**80**	**87**

Breakdown of 50 Java Examples

Type	# Projects	# Target Statements	# Inline Tests
Regex	15	17	17
String	15	20	20
Bit	16	25	26
Stream	2	2	2
Total	37	**64**	**65**

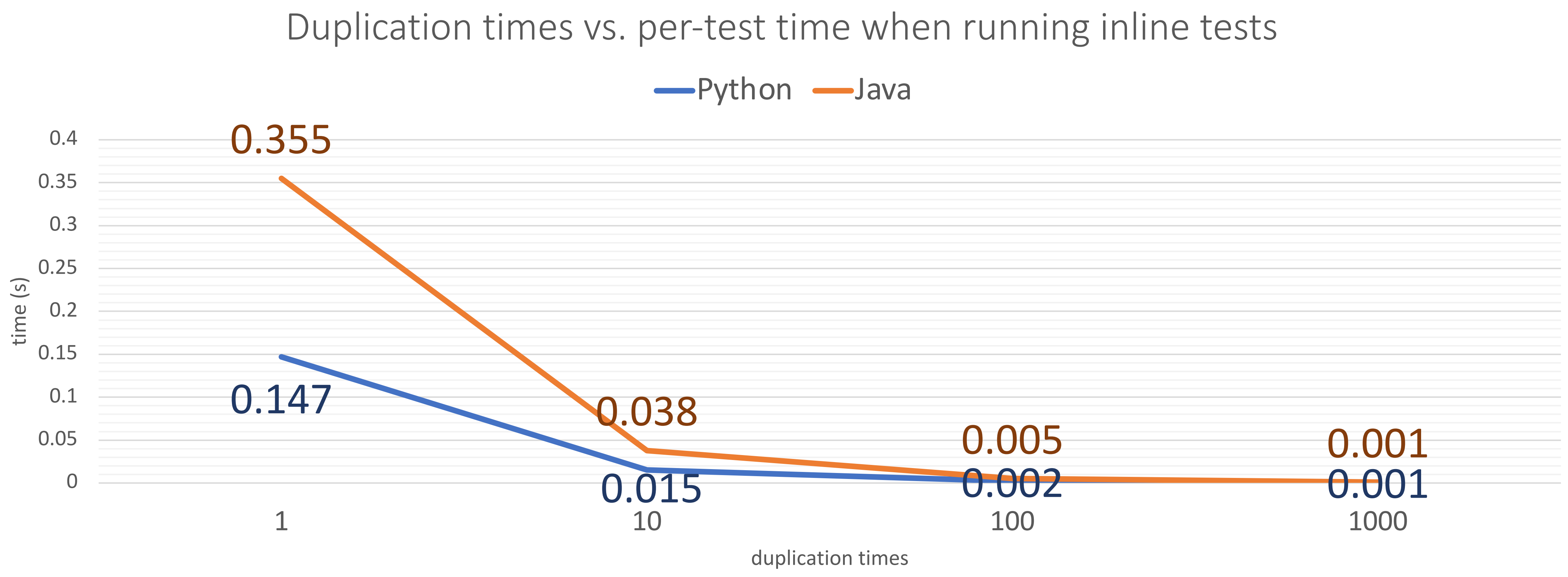
Evaluation Setup

- RQ1: How long does it take to run inline tests?
- RQ2: What is the runtime overhead when inline tests are enabled during the execution of existing unit tests?
- RQ3: What is the runtime overhead when inline tests are disabled during the execution of existing unit tests?

Experiment Setup

- Standalone experiments
  - Run inline tests alone
- Integrated experiments
  - Run inline tests and unit tests together
- Duplicating inline tests
  - Duplicate each inline test 10, 100 and 1000 times to simulate the costs as the number of inline tests grows

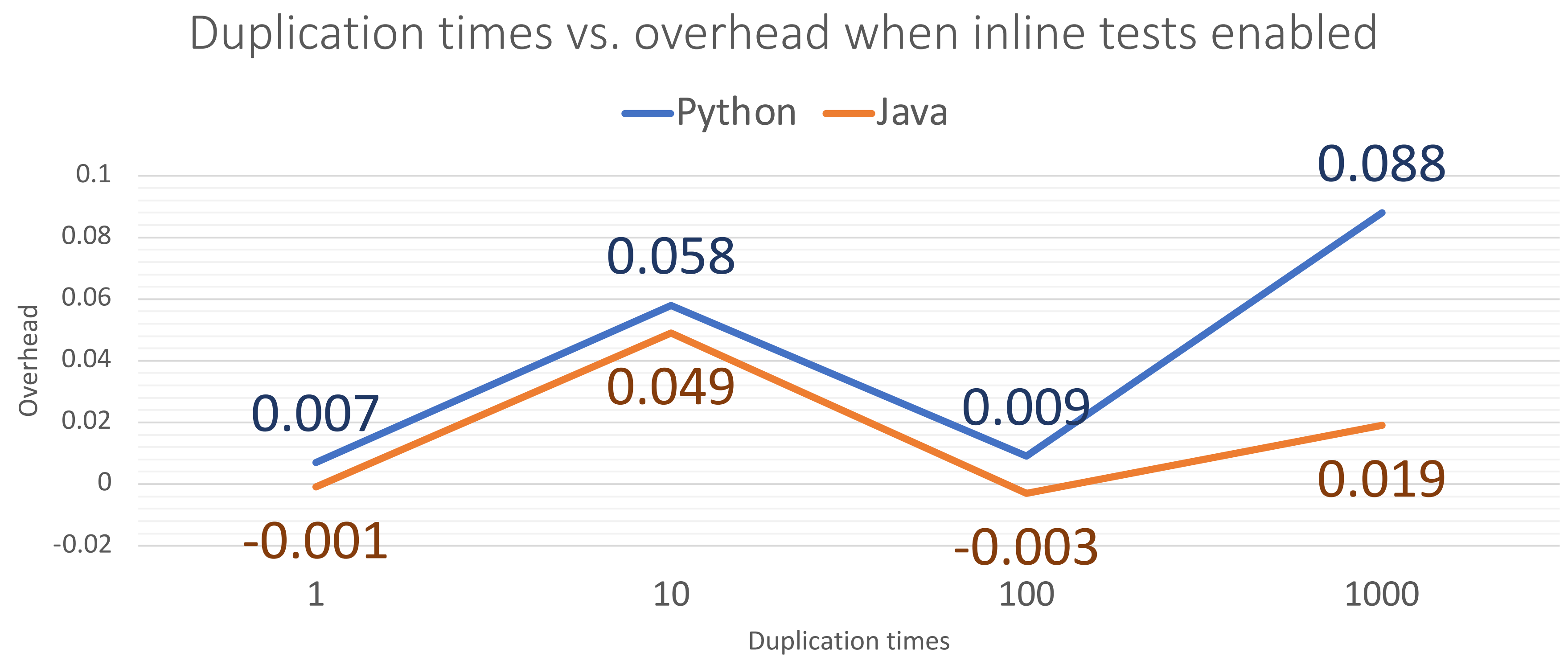
RQ1: How Long Does It Take to Run Inline Tests?



We run experiments on a machine with Intel Core i7-11700K @ 3.60GHz (8 cores, 16 threads)CPU, 64 GB RAM, and Ubuntu 20.04.

RQ2: What Is the Runtime Overhead When Inline Tests are Enabled During the Execution of Existing Unit Tests?

$$\frac{\textit{time of running unit tests with inline tests enabled} - \textit{time of running vanilla unit tests}}{\textit{time of running vanilla unit tests}}$$



We run experiments on a machine with Intel Core i7-11700K @ 3.60GHz (8 cores, 16 threads)CPU, 64 GB RAM, and Ubuntu 20.04.

User Study

- 1 tutorial with 3 examples
- 4 tasks of each type in Python
  - record time of understanding code and time of writing inline tests
- 1 survey with 4 questions
  - rate the difficulty of learning and writing inline tests
  - report their number of years of programming experience
  - say whether they think writing inline tests is beneficial
  - comment on how to improve I-Test
- 13 participants: 2 pilot studies, 9 valid responses
  - 6 graduate students, 2 undergraduate students and 1 professional software engineer

User Study Result

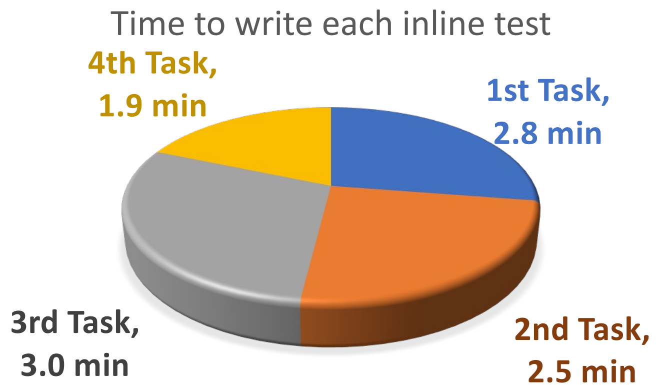
task 1: regular expression

task 2: string manipulation

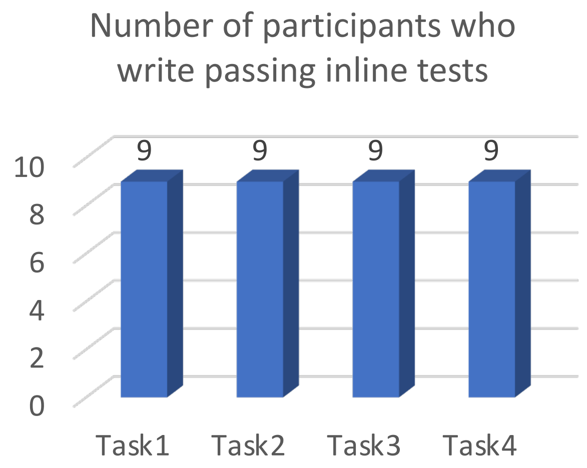
task 3: collection handling

task 4: bit manipulation

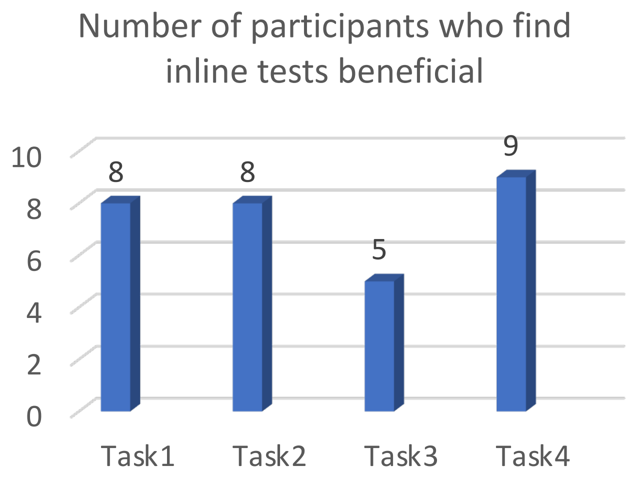
2.7 min on average



All participants can write passing inline tests



Participants usually find inline tests beneficial



Conclusion

- Introduce a new kind of tests, inline tests, perform statement-level testing
- Implement the first inline testing framework, I-Test
- Additional cost of inline testing is tiny
- Participants find it easy to learn and use inline testing

https://github.com/EngineeringSoftware/inlinetest

Yu Liu (Yuki): yuki.liu@utexas.edu