Proposal: CodeShovel::Python3

Lilli Freischem
The University of British Columbia
Vancouver, British Columbia, Canada
lilli.freischem@alumni.ubc.ca

Braxton J Hall
The University of British Columbia
Vancouver, British Columbia, Canada
braxtonhall@alumni.ubc.ca

Danhui Jia
The University of British Columbia
Vancouver, British Columbia, Canada
danhui.jia@alumni.ubc.ca

Brian Yeung
The University of British Columbia
Vancouver, British Columbia, Canada
brianyeung@live.com

ABSTRACT

"Take this shovel to dig in source code history for changes to specific methods and functions." [2] An online tool for finding artifacts of mutation in a git repository, CodeShovel appears to offer value in facilitating the excavation of information that can be lost to time in large repositories, yet it appears that the only material its blade can pierce is Java. We propose an extension of the CodeShovel tool to allow its use with the popular programming language Python 3.

ACM Reference Format:
Lilli Freischem, Braxton J Hall, Danhui Jia, and Brian Yeung. 2018. Proposal: CodeShovel::Python3. In *CPSC 311: Definition of Programming Languages.* ACM, New York, NY, USA, 3 pages. https://doi.org/00.0000/0000000.0000000

1 INTRODUCTION

"codeshovel [sic] is a tool for navigating how source code methods have evolved, across the kinds of evolutionary changes applied to the method, including most common refactorings, that the method saw throughout its life span. It is capable of tracking a method not only as it moves between line ranges, but as it moves through classes and around a codebase, from file to file." [6]

CodeShovel is a static analysis tool that navigates through a git commit history backward like a linked list, and builds an Abstract Syntax Tree (AST) for each file, and interprets it looking for modifications to a specified method. This can be useful for isolating changes to methods, assisting in predicting defects and explaining the story of how a method came to be.

While the use cases become immediately apparent for education, research, and industry use, CodeShovel is obviously restricted by its reliance on JavaParser.[1] Java may be popular, but it is by no means the only language in which software is written. With recent trends as well, it may not even be in the lead. [3]

[1]As seen in the CodeShovel repository, [2] the creation of the ASTs that CodeShovel uses to find pertinent data in a git history is done with the JavaParser package. [8]

Project Deliverable 3, Oct 18, 2018, Vancouver, BC
© 2018 Association for Computing Machinery.
ACM ISBN 000-0-0000-0000-0/00/00…$00.00
https://doi.org/00.0000/0000000.0000000

2 PROPOSITION

Python 3, cited as the most popular programming language today, [3] stands as a reasonable candidate for CodeShovel's first extension. As 84% of Python developers use Python primarily, [9] extending CodeShovel to the language would open it up to a vast number of possible new users.

Following this proposal's acceptance, the researchers aim to achieve the following four milestones:

- **MS1:** Evaluate the impact and feasibility of CodeShovel::Python3
- **MS2:** Derive a means to integrate a Python parser and AST with CodeShovel
- **MS3:** Implement CodeShovel::Python3
- **MS4:** Produce a final report of the efficacy of CodeShovel::Python3 along with a review of the researchers' process and the project as a whole

2.1 Effects

Should CodeShovel::Python3 be built, CodeShovel and its users would benefit from a larger set of repositories they could query for historical information about methods and functions.

For example, take the file `Battle.java` in the Java repository `gitinstance/JavaProject`.

- Let `shipOfTheseus()` be introduced at the commit `shajv00`
- Let `shipOfTheseus()` be moved to the file `Harbour.java` at the commit `shajv01`
- Let `Harbour.java` remain unmodified at the commit `shajv02`
- Let `shipOfTheseus()` be modified at the commit `shajv03`

```java
// shajv00
public class Battle {
    public void shipOfTheseus() {
        System.out.println("Off to battle!");
    }
}
```

```java
// shajv01
public class Harbour {
    public void shipOfTheseus() {
        System.out.println("Off to battle!");
    }
}
```

```java
// shajv03
public class Harbour {
    public int shipOfTheseus() { return 0; }
}
```

Let us also define semantically identical programs `Battle.py` and `Harbour.py` with the same semantic mutations in the Python 3 repository `gitinstance/pythonproject`.

- Let `ship_of_theseus()` be introduced at the commit `shapy00`
- Let `ship_of_theseus()` be moved to the file `Harbour.py` at the commit `shapy01`
- Let `Harbour.py` remain unmodified at the commit `shapy02`
- Let `ship_of_theseus()` be modified at the commit `shapy03`

```
# shapy00
class Battle:
  def ship_of_theseus():
    print("Off to battle!")
```

```
# shapy01
class Harbour:
  def ship_of_theseus():
    print("Off to battle!")
```

```
# shapy03
class Harbour:
  def ship_of_theseus():
    return 0
```

When CodeShovel examines `shipOfTheseus()` from the commit `HEAD` in `gitinstance/JavaProject`, it returns the results:

SHA	Change
shajv03	Return Type, Body
shajv01	Moved From File
shajv00	Introduced

Whereas when CodeShovel examines `ship_of_theseus()` from the commit `HEAD` in `gitinstance/pythonproject`, it returns:

ERROR

Following the construction of CodeShovel::Python3 as described in this proposal, the output of CodeShovel examining `shipOfTheseus()` from the commit `HEAD` in `gitinstance/JavaProject` remains the same.

However the output of CodeShovel examining `ship_of_theseus()` from the commit `HEAD` in `gitinstance/pythonproject` becomes:

SHA	Change
shapy03	Body
shapy01	Moved From File
shapy00	Introduced

3 MILESTONES

The bulk of our work will be divided into four milestones set internally, which are separate from the deliverables outlined in the project description.

3.1 MS1

Evaluate the impact and feasibility of CodeShovel::Python3

The researchers will begin by conducting an investigation into the prior work which they will be building from. The researchers aim to answer the following two research questions:

- **RQ1:** Can users of CodeShovel benefit from having a Python version of the tool?
- **RQ2:** What are use cases of CodeShovel::Python3 that are made measurably easier with the tool?

In addition, we aim to define a more fine grained approach to building the extension than the plan defined in this proposal.

To answer RQ1 and RQ2, the researchers will examine Felix Grund's CodeShovel thesis paper [7] for benefits and use cases of CodeShovel that translate directly to Python. In addition, the researchers will survey their peers to gauge their cohort's perceived use cases and benefits of having CodeShovel::Python3.

To deepen their technical understanding of CodeShovel, the researchers will as well fork the CodeShovel repository [2] to examine how it operates and come to understand the development process.

From here the researchers will have sufficient groundwork for developing the external requirements of CodeShovel::Python3. Further they will look to Python 3 AST generators and visitors in Java. One potential starting point is *ANTLR*, [1] which has a Java library which can be freely imported.

3.2 MS2

Derive a means to integrate a Python parser and AST with CodeShovel

Following the methodology of Grund (CodeShovel's creator) [7] the researchers will manually construct a set of unit tests that will also act as training data by which CodeShovel::Python3 will be tuned.

The researchers also expect to have rudimentary Python 3 parsing integration with the CodeShovel. This may be achieved with *ANTLR* [1] or another AST generating library to be determined following the delivery of MS1.

3.3 MS3

Implement CodeShovel::Python3

The full implementation of CodeShovel::Python3 will require interpretation of the AST generated for CodeShovel and fine tuning of various heuristics[2] until the unit tests created for MS2 pass.

So as to be able to interact with the tool online, the researchers will implement an online tool to use CodeShovel::Python3 hosted on cs310.students.cs.ubc.ca. Web service and UI repositories developed specifically for CodeShovel are publicly available, [4, 5] and can be forked and extended such that they can serve the CodeShovel::Python3 tool.

3.4 MS4

Produce a final report of the efficacy of CodeShovel::Python3 along with a review of the researchers' process and the project as a whole

[2] CodeShovel uses a combination of syntactic comparisons and lexical similarity functions with various weights to ascertain if two methods are "the same method." [7] These functions are tuned manually to the language until the training data is successfully classified by CodeShovel.

The final report shall include a review of the researchers' development up until the report's delivery. As well the report shall include a quantitative analysis of CodeShovel::Python3's accuracy and performance. At present the researchers foresee a third and final research question:

- **RQ3:** Are the results returned by CodeShovel::Python3 correct?

4 SCHEDULE

- **Background Research:** 2019/10/18 to 2019/10/30
  Delivery of MS1 on 2019/10/30
- **Prototype Development:** 2019/10/30 to 2019/11/13
  Delivery of MS2 on 2019/11/13
- **Iteration and Development:** 2019/11/13 to 2019/11/29
  To elicit feedback, the researchers will prepare a poster describing their work, and present it at a forum for CPSC 311 on 2019/11/29
- **Finalization:** 2019/11/29 to 2019/12/2
  Delivery of MS3 on 2019/12/2
  Delivery of MS4 on 2019/12/2

5 PROJECT VALIDITY

This proposal is for the UBC CPSC 311 Final Project. The researchers firmly believe that should the project be approved it would fulfill a 100% project as defined in the project specification. [10] Broadly, to achieve 100%, a project must support the delivery of the following Project Deliverables:

- **D4:** Background Research Report
  Met by the delivery of MS1
- **D5:** Project Proof-of-Concept and Plan
  Met by the delivery of MS2
- **D6:** Poster
  Met during the Iteration and Development period as described in 4
- **D7:** Final Project
  Met by the delivery of MS4

6 NEXT STEPS

As per 3.1, following the approval of this paper, the researchers will immediately proceed to working toward the delivery of MS1. This shall begin with Felix Grund's CodeShovel thesis paper, [7] forking `ataraxie/codeshovel`, [2] investigating *ANTLR*, [1] and developing a survey of their peers.

REFERENCES

[1] ANTLR. 2019. ANTLR. Retrieved October 17, 2019 from https://www.antlr.org
[2] ataraxie. 2019. CodeShovel. Retrieved October 17, 2019 from https://github.com/ataraxie/codeshovel
[3] Data Is Beautiful. 2019. Most Popular Programming Languages 1965 - 2019. Retrieved October 17, 2019 from https://www.youtube.com/watch?v=Og847HVwRSI
[4] braxtonhall. 2019. codeshovel-webservice. Retrieved October 17, 2019 from https://github.com/braxtonhall/codeshovel-webservice
[5] braxtonhall. 2019. codeshovel-webservice-ui. Retrieved October 17, 2019 from https://github.com/braxtonhall/codeshovel-ui
[6] codeshovel. 2019. codeshovel. Retrieved October 17, 2019 from http://cs310.students.cs.ubc.ca
[7] Felix Grund. 2019. CodeShovel : constructing robust source code history. Retrieved October 17, 2019 from https://open.library.ubc.ca/cIRcle/collections/ubctheses/24/items/1.0379647
[8] JavaParser. 2019. JavaParser - For processing Java code. Retrieved October 17, 2019 from https://javaparser.org
[9] JetBrains. 2018. Python Developers Survey 2018 Results. Retrieved October 17, 2019 from https://www.jetbrains.com/research/python-developers-survey-2018/
[10] Felipe Bañados Schwerter. 2019. CPSC 311 Final Project. Retrieved October 17, 2019 from https://www.students.cs.ubc.ca/~cs-311/current/assignment/project.html