CodeShovel: A Reusable and Available Tool for Extracting Source Code Histories

Felix Grund
Department of Computer Science
University of British Columbia
Vancouver, Canada
fgrund@cs.ubc.ca

Shaiful Chowdhury
Department of Computer Science
University of British Columbia
Vancouver, Canada
shaifulc@cs.ubc.ca

Nick C. Bradley
Department of Computer Science
University of British Columbia
Vancouver, Canada
ncbrad@cs.ubc.ca

Braxton Hall
Department of Computer Science
University of British Columbia
Vancouver, Canada
braxtonh@cs.ubc.ca

Reid Holmes
Department of Computer Science
University of British Columbia
Vancouver, Canada
rtholmes@cs.ubc.ca

***Abstract*—Being able to accurately understand how source code evolved is fundamentally important for both software engineers and researchers. Our ICSE 2021 Research Paper *CodeShovel: Constructing Method-Level Source Code Histories* describes a novel approach for quickly uncovering these method histories. The approach, codified in the CodeShovel tool, is available for researchers and practitioners alike to use and extend. It is *available* both as a public web service that can be used interactively or through a REST API and as a stand-alone Java component. This document details how to install and use CodeShovel, although all pertinent details are available online enabling CodeShovel to be *reused* as desired.**

***Index Terms*—software evolution, code histories, artifact**

CodeShovel Public Web Service

To try CodeShovel without installing any tools, use the public web service (also works with the REST API):

https://se.cs.ubc.ca/CodeShovel

I. CodeShovel

CodeShovel is a tool for identifying the complete history of a method at runtime by traversing the method's version history. Compared to traditional tooling like `git log` or standard IDE-based approaches, CodeShovel is robust to common source code transformations methods often undergo. These include the method being renamed or other signature changes, the method being extracted to another file, or the file containing the method being renamed or moved.

To start its analysis, CodeShovel clones the repository containing the method, although no pre-processing is performed beyond the standard clone. The developer then provides the required inputs and CodeShovel explores the history of the method and return results within a second or two.

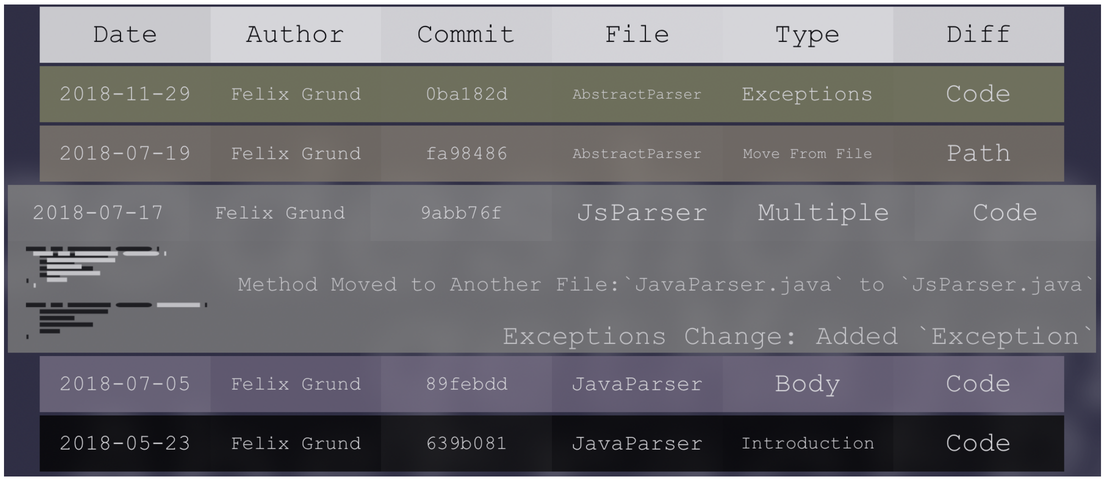
Inputs. CodeShovel requires several pieces of input to generate a history, although all are easily available to developers. Using the web service UI, only the method URL needs to be provided, all others are filled in by interactively clicking through the repository to select the method, although the REST web service and command line require these parameters explicitly:

- The URL for the repository containing the method.
- The path for the file containing the method.
- The name of the method.
- The line number for the method (used to differentiate overloaded methods).
- An optional SHA if the history desired should be explored from a point other than `HEAD`.

Outputs. In addition to the method change list, CodeShovel also performs a lightweight analysis of *how* the method evolved. Specifically, CodeShovel differentiates between the following kinds of changes:

- File move
- File rename
- Method move (extract method refactoring)
- Method body changes
- Method signature changes:
  - Method rename
  - Parameter list change
  - Parameter type change
  - Return type change
  - Exception change
  - Modifier change
- Method introduction

Every change will have one or more of these change kinds associated with it; for example, a method could be extracted from one file to another, be renamed, and have its visibility modifier changed all in a single commit. Being able to classify the change kind can make it easier for a developer to scan a longer list of changes and either identify the change they are interested in, or ignore the kinds of changes they are *not* interested in.



Date	Author	Commit	File	Type	Diff
2018-11-29	Felix Grund	0ba182d	AbstractParser	Exceptions	Code
2018-07-19	Felix Grund	fa98486	AbstractParser	Move From File	Path
2018-07-17	Felix Grund	9abb76f	JsParser	Multiple	Code
2018-07-05	Felix Grund	89febdd	JavaParser	Body	Code
2018-05-23	Felix Grund	639b081	JavaParser	Introduction	Code

```
Method Moved to Another File:`JavaParser.java` to `JsParser.java`
Exceptions Change: Added `Exception`
```

Fig. 1. The CodeShovel web interface applied to a file from CodeShovel's history. The row for the third change has been clicked on to show additional details and individual cells link back to the original version control system.

CodeShovel can return results in a rendered form (as can be seen in Figure 1) or as JSON for programmatic analysis. The JSON representation contains all output needed for tools to examine the change (including version control metadata and diffs), while the rendered output is more amenable to developer exploration. For example, all cells in the rendered view can be clicked to view the diff, the whole source file, or other relevant details in the version control system.

II. Using CodeShovel

There are three main versions of CodeShovel: a browser-based UI, a REST-based web service, and a command line client. Each of these can be installed locally, or a public version of the web service can be used. The instructions for each version are provided in `README.md`. CodeShovel is an open source project that is developed fully in the open and is licensed under the MIT Open Source License (see `LICENSE.md` for details).

Artifact Location

All source code, comprehensive documentation for installing and using CodeShovel, and evaluation oracles available online for use or extension:

https://github.com/ataraxie/codeshovel/

Web Service UI. The web UI provides a browser-based interface for interactively exploring method histories. This is the best version for a developer to use to explore the history of an individual method. The UI can be accessed either via the public web service, or by self-hosting the docker-based service. The public web service is best for evaluating CodeShovel, but it will be much more performant to self-host the service.

Web Service REST. The REST-based interface provides programmatic access to the backed via standard web APIs using any language. This is the best version to use for mining-style analyses where remote access might be required. For example, the history of a method can be accessed by:

```
curl "https://host/CodeShovel/getHistory? \
  gitUrl=${}&filePath=${}&methodName=${}& \
  startLine=${}&sha=${}"
```

Command Line. The command line interface is the quickest way to use the tool if you are using Java locally or are performing a single-machine mining repository project. This version can be invoked as follows:

```
java -jar codeshovel.jar \
  -repopath {rPath} -filepath {fPath} \
  -methodname {mName} -startline {lNum} \
  -sha {SHA} -outfile results.json
```

III. Artifact Documentation

A CodeShovel branch corresponding to the ICSE 2021 paper [1] (and this artifact) is available online [1,2], although most users will probably want to download and interact with more current branches (which will also contain the most up-to-date documentation).

All relevant details needed to install or use any of the three CodeShovel versions are described in `README.md`. Additional details pertinent to the containerized version can be found in `Dockerfile` and `docker-compose.yml`.

CodeShovel's source code is in `src/main` and its test infrastructure is in `src/test`. Language-specific oracle files are available in `src/test/resources/oracles`.

References

[1] F. Grund, S. Chowdhury, N. C. Bradley, B. Hall, and R. Holmes, "CodeShovel: Constructing method-level source code histories," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2021, pp. 1–13.

[1]https://github.com/ataraxie/codeshovel/tree/icse2021
[2]https://doi.org/10.5281/zenodo.4543820