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.