ArCode

(1.0)

A User Guide

Feb 2021

Table of Contents

Changes

Author	Date of Change	Description
Ali Shokri	02/05/2021	Version 1.0

Introduction

ArCode is a tool for analyzing programs written in Java, finding how APIs of a framework are being used in those programs, and providing recommendations on either how to correct possible API misuses or how to complete an incomplete program (w.r.t. APIs of that framework).

There are three main inputs to ArCode:

1. Framework jar file
2. The path to training projects' directory
3. The path to testing project's directory

ArCode leverages static analysis techniques to extract some facts about API dependencies and usage constraint from the framework. It then analyzes programs in the training repository to identify programs that are not violating API usage constraint. Mining these programs, it builds an API usage model, called Framework API Specification (FSpec) model. Finally, it analyzes testing projects to find how APIs are being used in them and provides recommendations on how to fix or complete (if needed) those programs.

If you are interested, you may find more details about ArCode from its technical paper.

Obtaining ArCode tool

In this section, we provide a we provide a step-by-step instruction to walk you through the process of building and using ArCode.

Use a released version

When we reach a stable version, we build the project and release it as a jar file. You can easily download the latest release from ArCode repository at GitHub.

Building ArCode from source code

In order to build a runnable version of ArCode from the source code, you need to have Maven3 installed on your machine. The link to Maven can be found from here.

Please use below command to build the ArCode runnable jar file once you have installed maven:

```
$ mvn clean package
```

If the process finishes with success, ArCode jar file (e.g. arcode-1.0-SNAPSHOT.jar) is created and is ready to be used.

Running ArCode

For running ArCode, you may use the following template in the command line. In the next section, a tutorial of runing the tool is provided.

```
$ java -jar ARCODE_JAR_FILE -framework FRAMEWORK_OF_INTEREST -fspecOutputPath
PATH_TO_FSPEC_OUTPUT -trainProjectsPath PATH_TO_TRAINING_PROJECTS -
testProjectsPath PATH_TO_TESTING_PROJECTS -exclusionFilePath
PATH_TO_EXCLUSION_FILE -frameworkJarPath PATH_TO_FRAMEWORK_JAR_FILE -
frameworkPackage PACKAGE_NAME_OF_FRAMEWORK
```

ARCODE_JAR_FILE: Path to ArCode jar file that you downloaded from [releases](../../releases/) or built from ArCode source code using Maven.

FRAMEWORK_OF_INTEREST: The framework that you would like to extract and create a model of its API usages in training projects. This model is called Framework API Specification model (FSpec). ArCode also performs static analysis to find dependencies between APIs inside the framework jar file and creates an Inter-framewrok Dependency model (IFD). You may find more details about the approach from ArCode [paper](https://a-shokri.github.io/assets/Ali%20Shokri-ArCode-ICSA2021-Accepted.pdf).

PATH_TO_FSPEC_OUTPUT: Created FSpec will be serialized and saved in this directory. ArCode will later restore this FSpec and use it in the recommendation phase.

PATH_TO_TRAINING_PROJECTS: Path to the bytecode (jar files) of the training projects that are incorporating the framework of interest. ArCode analyzes these programs to create a Graph-based API Usage Model (GRAAM) for each program. Then, performing graph-based operations, it creates the framework's FSpec.

PATH_TO_TESTING_PROJECTS: Path to the bytecode (jar files) of the testing projects that are incorporating the framework of interest. ArCode analyzes these programs to create a Graph-based API Usage Model (GRAAM) for each program. The, leveraging the previously created FSpec, it identifies API misuses or APIs that are needed to be added to the program to make the program a correct implementation of an architectural tactic or a pattern.

PATH_TO_EXCLUSION_FILE: This is a file entitled as "JAASJavaExclusions.txt" in the config folder of ArCode source code. In the current version of ArCode, you need to have that file on your system and introduce its path to ArCode. This file specifies packages to be excluded from program analysis process. We need to introduce these packages to avoid explosion of call-graph creation in that process.

PATH_TO_FRAMEWORK_JAR_FILE: Path to jar file of the framework of interest.

PACKAGE_NAME_OF_FRAMEWORK: Since there might be more than one framework in a jar file, we introduce the packaging of the framework through this property.

Tutorial

To find a practical understanding of ArCode please watch this tutorial video.

Contact

Please do not hesitate to reach out to us should you have any question about ArCode. You may get in contact with us through as8308@rit.edu.