Semantic Patches for Java Program Transformation (Artifact)

Hong Jin Kang
School of Information Systems, Singapore Management University, Singapore
hjkang.2018@phdis.smu.edu.sg

Ferdian Thung
School of Information Systems, Singapore Management University, Singapore
ferdiant.2013@phdis.smu.edu.sg

Julia Lawall
Sorbonne Université/Inria/LIP6, France
Julia.Lawall@lip6.fr

Gilles Muller
Sorbonne Université/Inria/LIP6, France
Gilles.Muller@lip6.fr

Lingxiao Jiang
School of Information Systems, Singapore Management University, Singapore
lxjiang@smu.edu.sg

David Lo
School of Information Systems, Singapore Management University, Singapore
davidlo@smu.edu.sg

Abstract

The program transformation tool Coccinelle is designed for making changes that is required in many locations within a software project. It has been shown to be useful for C code and has been been adopted for use in the Linux kernel by many developers. Over 6000 commits mentioning the use of Coccinelle have been made in the Linux kernel.

Our artifact, Coccinelle4J, is an extension to Coccinelle in order for it to apply program transformations to Java source code. This artifact accompanies our experience report "Semantic Patches for Java Program Transformation", in which we show a case study of applying code transformations to upgrade usage of deprecated Android API methods to replacement API methods.

2012 ACM Subject Classification Software and its engineering → Software notations and tools

Keywords and phrases Java, semantic patches, automatic program transformation

Digital Object Identifier 10.4230/DARTS.5.2.10

Acknowledgements This research was supported by the Singapore National Research Foundation (award number: NRF2016-NRF-ANR003) and the ANR ITrans project.

Related Article Hong Jin Kang, Ferdian Thung, Julia Lawall, Gilles Muller, Lingxiao Jiang, and David Lo, "Semantic Patches for Java Program Transformation", in 33rd European Conference on Object-Oriented Programming (ECOOP 2019), LIPIcs, Vol. 134, pp. 22:1–22:27, 2019.
https://dx.doi.org/10.4230/LIPIcs.ECOOP.2019.22

Related Conference 33rd European Conference on Object-Oriented Programming (ECOOP 2019), July 15–19, 2019, London, United Kingdom

Dagstuhl Artifacts Series, Vol. 5, Issue 2, Artifact No. 10, pp. 10:1–10:3
Dagstuhl Artifacts Series
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Scope

In this document, instructions to set up Coccinelle4J are provided. Furthermore, we provide a selection of semantic patches that can be applied by Coccinelle4J to source code extracted from real-world Java projects. These semantic patches are written in SmPL, a scripting language provided by Coccinelle [1].

2 Content

The artifact package includes:

- a Dockerfile to build the Docker image `coccinelle4j/coccinelle4j`
- a document that provides instructions on how to run Coccinelle4J (ecoop-artifact.pdf)
- Coccinelle4J's source code
- The examples described in the experience report. For each example, we include
  - semantic patch specified in SmPL
  - some .java source files extracted from real-world Java projects
  - output of each semantic patch after applying it with Coccinelle4J

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). To minimize setup problems, we also provide a Docker image.

3.1 Docker

A Docker image is similar to a virtual machine image, simplifying the set up of a project's environment. However, unlike a virtual machine, Docker containers are lightweight, sharing the operating system's kernel with the host machine.

We use Docker to run Coccinelle4J in a container so that the dependencies of Coccinelle4J can be installed in an environment isolated from the rest of the machine. We provide a Docker image `coccinelle4j/coccinelle4j:ecoop` to easily set up containers that already have Coccinelle4J installed. This image also contains the examples described in the experience report.

The instructions to install Docker varies between operating systems and can be found on the official Docker document at `https://docs.docker.com/install/overview/`.

With Docker installed, the following commands can be executed to create a container based on our Docker image. We have uploaded the image at DockerHub and Docker will automatically fetch the coccinelle4j image from DockerHub. This image is approximately 3.54GB.

```
docker pull coccinelle4j/coccinelle4j:ecoop
docker run -it coccinelle4j/coccinelle4j:ecoop /bin/bash
```

The command will start a new container of the coccinelle4j image and run `bash` on it. On some machines, executing the above commands as root may be required.

3.2 Make

If Docker is unavailable, an alternative to set up Coccinelle4J is to build the Coccinelle4J executable using make. OCaml (with a version >4.04), git, autoconf, make should be installed first.

```
git clone https://github.com/kanghj/coccinelle
cd coccinelle
git checkout java
./autogen && ./configure
make && sudo make install
```

4 Tested platforms

In general, Coccinelle4J is supported on any Unix-like platform. The Docker image we have provided should work on any platform supporting Docker.

5 License

The artifact is available under GNU GPL version 2.

6 MD5 sum of the artifact

58763d6c633d1cc93c2ed3fd76e75960

7 Size of the artifact

The size of the zip file is 101.1MB. The size of the docker image is about 3.5GB

References

1 Yoann Padioleau, Julia L Lawall, and Gilles Muller. SmPL: A domain-specific language for specifying collateral evolutions in Linux device drivers. *Electronic Notes in Theoretical Computer Science*, 166:47–62, 2007.