

Singapore Management University

## Institutional Knowledge at Singapore Management University

---

Research Collection School Of Computing and Information Systems

School of Computing and Information Systems

---

7-2019

### Semantic patches for Java program transformation (artifact)

Hong Jin KANG

Singapore Management University, [hjkang.2018@phdcs.smu.edu.sg](mailto:hjkang.2018@phdcs.smu.edu.sg)

Thung Ferdian

Singapore Management University, [ferdianthung@smu.edu.sg](mailto:ferdianthung@smu.edu.sg)

Julia LAWALL

Sorbonne Universite, Inria LIP6

Gilles MULLER

Sorbonne Universite, Inria LIP6

Lingxiao JIANG

Singapore Management University, [lxjiang@smu.edu.sg](mailto:lxjiang@smu.edu.sg)

*See next page for additional authors*

Follow this and additional works at: [https://ink.library.smu.edu.sg/sis\\_research](https://ink.library.smu.edu.sg/sis_research)



Part of the [Programming Languages and Compilers Commons](#), and the [Software Engineering Commons](#)

---

#### Citation

KANG, Hong Jin; Ferdian, Thung; LAWALL, Julia; MULLER, Gilles; JIANG, Lingxiao; and LO, David. Semantic patches for Java program transformation (artifact). (2019). *33rd European Conference on Object-Oriented Programming (ECOOP 2019): London, July 15-19: Proceedings*. 5, 10:1-3.

Available at: [https://ink.library.smu.edu.sg/sis\\_research/4813](https://ink.library.smu.edu.sg/sis_research/4813)

This Conference Proceeding Article is brought to you for free and open access by the School of Computing and Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Computing and Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email [cherylids@smu.edu.sg](mailto:cherylids@smu.edu.sg).

---

**Author**

Hong Jin KANG, Thung Ferdian, Julia LAWALL, Gilles MULLER, Lingxiao JIANG, and David LO

# 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 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.3.2.1

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

**Related Article** Placeholder

**Related Conference** 42nd Conference on Very Important Topics (CVIT 2016), December 24–27, 2016, Little Whinging, United Kingdom

## 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].



© Hong Jin Kang, Ferdian Thung, Julia Lawall, Giles Muller, Lingxiao Jiang, David Lo; licensed under Creative Commons Attribution 3.0 Germany (CC BY 3.0 DE)

*Dagstuhl Artifacts Series*, Vol. 3, Issue 2, Artifact No. 1, pp. 1:1–1:3



DAGSTUHL  
ARTIFACTS SERIES

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1:2 Semantic Patches for Java Program Transformation: (Artifact)

### 6 **2** Content

7 The artifact package includes:

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

### 15 **3** Getting the artifact

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

#### 19 **3.1** Docker

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

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

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

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

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

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

#### 38 **3.2** Make

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

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

#### 48 **4** Tested platforms

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

#### 51 **5** License

52 The artifact is available under GNU GPL version 2.

#### 53 **6** MD5 sum of the artifact

54 58763d6c633d1cc93c2ed3fd76e75960

#### 55 **7** Size of the artifact

56 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.