

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.VOL.ISS.ART](https://doi.org/10.4230/DARTS.VOL.ISS.ART)

Related Article Hong Jin Kang, Ferdian Thung, Julia Lawall, Giles Muller, Lingxiao Jiang, David Lo, "Semantic Patches for Java Program Transformation", in Proceedings of the 32nd European Conference on Object-Oriented Programming (ECOOP 2018)

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.

2 Content

The artifact package includes:



© 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. [VOL](#), Issue [ISS](#), Artifact No. [ART NO.](#), pp. [ART:1–ART:5](#)



DAGSTUHL ARTIFACTS SERIES

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

XX:2 Semantic Patches for Java Program Transformation: (Artifact)

- 8 ■ a Dockerfile to build the Docker image `coccinelle4j/coccinelle4j`
- 9 ■ this document that provides instructions on how to run Coccinelle4J
- 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

3 Getting the artifact

16 There are two methods to set up Coccinelle4J. To minimize setup problems, it is preferable to use
17 the first method using Docker.

3.1 Docker

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

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

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

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

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

35 The command will start a new container of the `coccinelle4j` image and run `bash` on it. On
36 some machines, executing the above commands as root may be required. Next, the instructions to
37 run the examples are in Section 3.3.

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  
42 git clone https://github.com/kanghj/coccinelle  
43 cd coccinelle  
44 git checkout java  
45 ./autogen && ./configure  
46 make && sudo make install  
47
```

3.3 Instructions for running examples

49 In our paper, we provided 7 examples of deprecated Android API methods, and show semantic
50 patches that can be used to migrate them to their corresponding replacement API methods.

We provide these examples both in the docker image and in the git repository hosted at <https://github.com/kanghj/coccinelle>.

Each example is contained in sub-directory in the `ecoop_example_patches` directory. Every sub-directory consists of the semantic patch (a `.cocci` file) and examples of source code (`.java` files) that the semantic patch will be matched on. Some sub-directories may contain a file with isomorphisms that we extracted from the Java projects the source files were taken from. More details about each example are described in the experience report.

To apply the semantic patches on the source file, the `spatch` command can be executed within each sub-directory. For the first example of replacing `sendStickyBroadcasts`, corresponding to Listing 11 in the experience report, the following command can be run.

```
cd ecoop_example_patches/sticky_broadcasts
spatch --sp-file sticky_broadcasts.cocci FileDownloader.java
```

The first example can be found in the `sticky_broadcasts` directory (which is under the directory `ecoop_example_patches/`). `spatch` takes a semantic patch (in this case, `sticky_broadcasts.cocci`, specified with the `-sp-file` argument) and source files (in this case, `FileDownloader.java`) as input. By default, the patch generated by Coccinelle4J is printed to standard output. If it is desirable for Coccinelle4J to modify the source file directly, the `-in_place` flag can be passed to `spatch`. Apart from the patch generated, Coccinelle4J will print other details to standard output. For example, Coccinelle4j will print a list of files that it skipped. These files are skipped because they do not contain tokens necessary for a successful match against the semantic patch.

For the example of `sticky_broadcasts.cocci`, Coccinelle4J produces the following output (printed to standard output) after the command above is executed.

```
HANDLING: FileDownloader.java
diff =
--- FileDownloader.java
+++ /tmp/cocci-output-4196-ae14a8-FileDownloader.java
@@ -677,7 +677,7 @@ public class FileDownloader extends Serv
     end.putExtra(EXTRA_LINKED_TO_PATH, unlinkedFromRemotePath);
     }
     end.setPackage(getPackageName());
-    sendStickyBroadcast(end);
+    sendBroadcast(end);
     }

@@ -695,7 +695,7 @@ public class FileDownloader extends Serv
     added.putExtra(EXTRA_FILE_PATH, download.getSavePath());
     added.putExtra(EXTRA_LINKED_TO_PATH, linkedToRemotePath);
     added.setPackage(getPackageName());
-    sendStickyBroadcast(added);
+    sendBroadcast(added);
     }

/**
```

The instructions to apply the semantic patches for the other examples are as follows. From the `set_text_size` directory, run the following command. This applies the semantic patch described in Listing 13 in the experience report. The `-iso` option passes in a file containing isomorphisms for the project.

XX:4 Semantic Patches for Java Program Transformation: (Artifact)

```
102 spatch --sp-file set_text_size.cocci \  
103 --iso lucid_browser.iso CustomWebView.java
```

104 From the `get_color` directory, run the following command. This applies the semantic patch
105 described in Listing 17 in the experience report.

```
106 spatch --sp-file get_color.cocci PushNotifications.java
```

107 From the `should_vibrate` directory, run the following command. This applies the semantic
108 patch described in Listing 19 in the experience report.

```
109 spatch --sp-file should_vibrate.cocci IncomingRinger.java
```

110 From the `get_height` directory, run the following command. This applies the semantic patch
111 described in Listing 22 in the experience report.

```
112 spatch --sp-file get_height.cocci TouchUtils.java
```

113 From the `on_console_message` directory, run the following command. This applies the semantic
114 patch described in Listing 25 in the experience report.

```
115 spatch --sp-file on_console_message.cocci ViewFileFragment.java
```

116 From the `get_drawable` directory, run the following command. This applies the semantic patch
117 described in Listing 26 in the experience report.

```
118 spatch --sp-file get_drawable.cocci \  
119 DisplayUtils.java SimpleListItemDividerDecoration.java
```

120 In total, 7 examples are provided. A script `run_examples.sh` that runs all the examples is
121 included in the `ecoop_example_patches/` directory. Instead of manually running each example
122 one by one, executing `run_examples.sh` will apply all the semantic patches mentioned above on
123 the examples and write them into `output.patch` in each directory.

124 3.4 Running Coccinelle4J on an entire project

125 We provide examples to run Coccinelle4J on entire projects. Each directory contains a script
126 (`download_project.sh`) to download a project that a patch can be run on, and checks out the
127 version of the project used in our experience report.

128 Run the following command in the root of the `ecoop_example_patches` directory to clone
129 the projects. This command will clone the relevant projects into the directories containing the
130 example patches.

```
131 ./download_all_projects.sh
```

132 Run the following command in the root of the `ecoop_example_patches` directory to run the
133 semantic patches on the projects. This produces a `project.patch` file in every directory under
134 `ecoop_example_patches`, excluding `get_drawable`. We omit running `get_drawable.cocci` since
135 it was used an example only to demonstrate the limitations of Coccinelle4J in our experience
136 report. The `project.patch` files contains all the additions and deletions Coccinelle4J generated
137 for each project.

```
138 ./run_examples_on_entire_projects.sh
```

To count the number of lines in a semantic path, `grep` is used. For example, the following command can be executed (in the `sticky_broadcasts` directory) to count the number of non-empty lines in the `sticky_broadcasts.cocci`.

```
grep -cve "\s*$" sticky_broadcasts.cocci
```

To count the number of additions and deletions in the diff generated by Coccinelle4J, the following command can be executed. The command looks for lines starting with "-" or "+" in the diff. This command should be run in any of the directories (e.g. `sticky_broadcasts`)

```
grep -ce "[+-]\s" project.patch
```

A similar command can be used to count the number of files modified by Coccinelle4J.

```
grep -ce "diff" project.patch
```

For an alternative to executing the commands above for counting the number of lines/files in the semantic patch or the patch generated by Coccinelle4J, a convenience script `print_data.sh` can be used instead.

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 Size of the artifact

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