Electronic Communications of the EASST
Volume 63 (2014)

Proceedings of the Eighth International Workshop on Software Clones (IWSC 2014)

Measuring Copying of Java Archives

— Position Paper —

Tetsuya Kanda, Daniel M. German, Takashi Ishio and Katsuro Inoue

6 pages

Guest Editors: Nils Göde, Yoshiki Higo, Rainer Koschke
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Measuring Copying of Java Archives

Tetsuya Kanda[1], Daniel M. German[2,1], Takashi Ishio[1] and Katsuro Inoue[1]

[1] t-kanda@ist.osaka-u.ac.jp, ishio@ist.osaka-u.ac.jp, inoue@ist.osaka-u.ac.jp
Graduate School of Information Science and Technology
Osaka University, Japan

[2] dmg@uvic.ca
Department of Computer Science
University of Victoria, Canada

Abstract: Copying the whole of a library is one of the major types of reuse in software development. In Java, a single library archive file often contains other libraries it depends on, but users of the library hardly know about such inner libraries. Since reusing libraries is a black-box method, developers may combine some libraries without knowing that those libraries contain the same library inside independently. As a result, a library may contain inside several copies of a library it reuses. In this research, we measured copying of jar archives in the Maven Central Repository, a collection of open source Java libraries. Our results show that about 14% of top-level jar files are reused in other jar files and some of them are duplicated in a single jar file. We also found that some libraries contain two or more different versions of the same library.

Keywords: software reuse, Java library, duplication

1 Introduction

Reusing software components reduces time and cost when constructing new software, and copying the whole of a library into the software development project is one of the major types of reuse. Heinmann *et al.* showed that software reuse is common among open source Java projects and the black-box is the predominant form of reuse [HDG+11].

In the case of Java, library archive files often contain their dependent libraries. One reason is that developers want to use specific versions of libraries that might be considered reliable.

Once black-box reuse method has been done, it might not be known which version of which library is included in the library archive file. Davis *et al.* pointed out that the provenance of included components is not clearly stated and they proposed a method to determine the provenance of source code contained within Java archives [DGGH13].

However, there is a possibility that developers are also copying duplicated libraries in the reused libraries without knowing that. When developers copy some libraries into their project, they may also unconsciously copy the same version of the library they already have or copy different versions of the library, but developers might not be aware of this duplication.

The mainstream of the software clone research is for the source code [Kos07] and few researches focuses clones of other software artifacts. There are some researches for Java archives

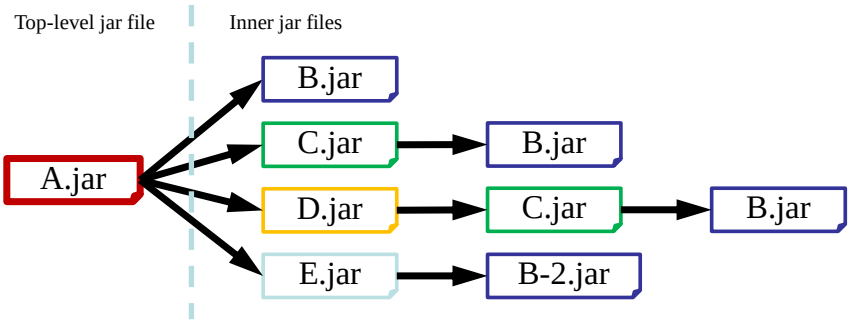

Figure 1: Example of nested jar files

[BHV98, SMBZ07]. They dealt with a problem that how to compress class files to reduce file size, but they paid no attention to the duplication of class files or whole of archives.

In this research, we performed an experiment to measure copying of jar archives in the Maven Central Repository, a collection of open source Java libraries. We set these research questions as a first step of the study of this type of duplication.

RQ1: How many jar files in a large software repository contain jar files inside and how many jar files are reused?

RQ2: Does duplication of reused jar files in other jar files really exist? If so, are those duplicated jar files the same version or different versions?

2 Background

Apache Maven [Mav] is a software project management and comprehension tool. It automatically downloads dependent Java libraries from Maven repositories at build time. Maven Central Repository (Maven2) is the default repository of Apache Maven. Maven2 repository contains many popular libraries and projects.

Java archive file is the typical format used to distribute Java applications and libraries. A Jar file contains Java class files and metadata and resources, and even another jar archive inside.

We define the term “top-level jar file” and “inner jar file” in this paper. A “top-level jar file” is a jar file found in the Maven2, and therefore, it corresponds to a component ready to be reused. An “inner jar file” is a jar file that is included in another jar file, either a “top-level jar file” or an “inner jar file”.

Figure 1 shows an example of a library with nested jar files. A node corresponds to a jar file. The jar file at the start of the arrow contains the jar file at the end of the arrow. In this case, the top-level jar file *A.jar* is found in the target repository and contains four inner jar files in it; *B.jar*, *C.jar*, *E.jar*. *C.jar* contains *B.jar* which is exactly same file as *B.jar* under *A.jar*. *D.jar* contains *C.jar* so *B.jar* appeared again inside of *C.jar*. *E.jar* contains *B-2.jar* which is the newer version of *B.jar*. In Figure 1, all jar files in the right side of *A.jar* are inner jar file of *A.jar*. *B.jar* and *C.jar* are duplicated, and there are two versions of *B.jar* (*B.jar* and *B-2.jar*).

Table 1: Example of how the jar filename was use to identify the name of the library

Path	Filename	Detected library name
/maven/org/geoserver/web/**1.4.0-RC3/**	web-1.4.0-RC3.jar	web
/maven/org/apache/archiva/archiva/**1.1/**	archiva-1.1-src.jar	archiva-src

3 The experiment

We conducted an experiment to find how many archive files contained duplicate archive files inside. We detected two types of duplication of jar files: the same version of the same library and the different versions of the same library.

3.1 Setup

We used a framework for Software Bertillonage proposed by Davis *et al.* [DGGH13]. The framework extracts metrics of source and archive files. We use two metrics, the filename and SHA1 hash of the file contents, to find jar files with exactly the same contents. If the file is contained in the archive file, SHA1 hash of the parent file is also extracted so that we can find out the contents of jar file.

3.2 Inner Jar Files

There are 607,319 top-level jar files in the Maven2 repository. Removing exactly the same files, with the same file name and the same hash, we get 599,498 top-level jar files. Checking the contents inside each top-level jar files, we found that 4,747 top-level jar files contain at least one jar file inside. 1,833 of them contains only one jar file and the largest one has 282 jar files in it, 13.1 on average and median was 2. We also found that 118,361 different inner jar files are contained in other jar files and 89,054 of them are found in the Maven2 repository as a top-level jar file. This means that most inner jar files are reused directly from the Maven2 repository.

3.3 Detecting Duplication

To find the two types of duplication inside jar files, we checked inner jar files using the following method:

First, we identify duplication of the same version of the libraries. If two jar files have the same file name and the same file hash, this means that they have exactly the same contents so they are considered as duplicated and they are the same version. We did not care about the nest level of jar file. In Figure 1, three *B.jar* are all different nest level counting from *A.jar*, but it does not affect the analysis.

Second, we identify duplication of different versions of the libraries. To detect different versions of the same library, we remove the version information from the jar file name. Version names are not only restricted in the number but also some strings such as “RC” and “SNAPSHOT”. We found that many libraries are also found in the Maven2 repository so we use the jar path name in Maven2 to identify its version. In the Maven2, most projects have their own

Table 2: Analysis result for A.jar in Figure 1

Step	File list
Unique inner jar file	B, C, D, E, B-2
Unique inner jar file without version names	B, C, D, E

Table 3: Duplication of inner jar files

	Contains inner jar	Duplication Type			Total duplication
		Same	Different	Both	
#files	4,747	105	394	30	469
#projects	886	39	49	14	73

directory, and a subdirectory for each version. We regard the directory name as the version name of the library and remove it from file name of the library. We also remove a leading hyphen or underscore with the version name. Table 1 shows two examples. This step is skipped if the library is not found in the Maven2 repository since we cannot get the version name from the directory name.

Table 2 shows the example resuly of analysis for Figure 1. In the example Figure 1, *B.jar* appears three times and *C.jar* appears twice. In this case *B.jar* and *B-2.jar* have the same library name so they are determined as different versions of the library B.

Table 3 shows the results of the experiment. We count the number of libraries in two ways; counting number of jar files and counting number of projects used disregarding their version as described as Step 3.

In total, 469 jar files contain duplicate libraries inside, about 10% of the top-level jar files that contains inner jar files. Counting the number of projects, the result also shows that about 8% of maven projects contain inner jar files that have duplicated libraries in them.

We found both types of duplication in the Maven2 repository: 394 jar files contain the same version of the same library and 105 jar files contain the different versions of the same library. We also found that 30 files have both types of duplication.

Some jar files which have duplication of different versions of the archive files have “test” in their file name. The inner jar files of *nexus-app-1.7.1-tests.jar*, listed in Table 4, it contains 28 different inner jar files, including six different versions of log4j library. In total there are 32 inner jar files named log4j inside *nexus-app-1.7.1-tests.jar* and each versions of log4j appeared 3 to 7 times.

3.4 Revisiting Research Questions

RQ1 *How many jar files in a large software repository contain jar files inside and how many jar files are reused?*

In the Maven2 repository, there are 4,747 of 599,498 jar files that contain inner jar files. The number of inner jar files is at least one and at most 282 files, 13.1 on average and median was 2. From the point of view of reuse, 89,054 of top-level jar files in the Maven2 repository also appeared as inner jar files.

Table 4: List of inner jar files of nexus-app-1.7.1-tests.jar

antlr-2.7.6 (7)	nexus-3148-1.0.20100111.064938-1	nexus-indexer-1.0-beta-5-20080718.231118-50
antlr-2.7.7 (5)	nexus-3148-1.0.20100111.065026-2	nexus-indexer-1.0-beta-5-20080730.002543-149
log4j-1.2.12 (5)	nexus-indexer-1.0-beta-3-20010711.162119-2	nexus-indexer-1.0-beta-5-20080731.150252-163
log4j-1.2.13 (5)	nexus-indexer-1.0-beta-3-SNAPSHOT	nonuniquesnap-1.1-SNAPSHOT
log4j-1.2.13-sources (5)	nexus-indexer-1.0-beta-4	plexus-plugin-manager-1.0-20081125.071530-1
log4j-1.2.14 (5)	nexus-indexer-1.0-beta-4-SNAPSHOT	sonatype-test-evict_1.4_mail-1.0-SNAPSHOT
log4j-1.2.14-sources (5)	nexus-indexer-1.0-beta-4-SNAPSHOT-cli	very.very.long.project.id-1.0.0-20070807.081844-1
log4j-1.2.15 (3)	nexus-indexer-1.0-beta-4-SNAPSHOT-jdk14	very.very.long.project.id-1.1-20070807.081844-1
log4j-1.2.8 (7)	nexus-indexer-1.0-beta-4-SNAPSHOT-sources	
log4j-1.2.9 (7)	nexus-indexer-1.0-beta-5-20080711.162119-2	

(n) represents the number of files

RQ2 *Does duplication of reused jar files in other jar files really exist? If so, are those duplicated jar files the same version or different versions?*

Yes, 10% of jar files which have inner jar files contains duplicated jar files. We can say that the duplication in libraries are not an unusual problem. Both type of duplication are found in the Maven2 repository.

4 Conclusion and Future Work

Developers reuse existing libraries by copying them into the software development project and this style reuse reduces time and cost on constructing new software. On the other hand, there is a possibility that developers are also copying duplicated libraries in the reused libraries without knowing that.

The result of our experiment indicates that the duplication of archive files in a single archive file is not frequent, but it exists. And furthermore, we must remember that many archive files are copied into others so that further duplication can occur. Concretely, we found that about 5,000 jar files in the Maven2 repository contain other jar files in them and about 470 of them contains duplicate libraries, some of them are the same version and some of them are different versions. We also found that about 14% of top-level jar files in the Maven2 repository are copied into other top-level jar files.

Based on this result, we are planning to perform further studies. We found duplication of jar files but did not check all contents of them, and finding out which duplicated archive is most frequently reused is our future work. In addition, we should also analyze other types of archive files. We only used jar archives but the Maven2 repository has .zip, .tar.gz, .war, .ear formats of archives and these are not limited in binary archives but also source archives.

Another interesting fact is that there are some inner jar files and some duplications even though Apache Maven has a system to download needed jar files at built time. We want to investigate whether it is possible to remove the duplication.

Acknowledgements: This work is supported by Japan Society for the Promotion of Science, Grant-in-Aid for Scientific Research (S) “Collecting, Analyzing, and Evaluating Software Assets for Effective Reuse” (No.25220003). This work is also supported by Osaka University Program for Promoting International Joint Research, “Software License Evolution Analysis.”

Bibliography

[BHV98] Q. Bradley, R. N. Horspool, J. Vitek. JAZZ: An Efficient Compressed Format for Java Archive Files. In *Proceedings of the 1998 Conference of the Centre for Advanced Studies on Collaborative Research*. CASCON '98, pp. 7–. 1998.

[DGGH13] J. Davies, D. M. Germán, M. W. Godfrey, A. Hindle. Software Bertillonage - Determining the provenance of software development artifacts. *Empirical Software Engineering* 18(6):1195–1237, 2013.

[HDG+11] L. Heinemann, F. Deissenboeck, M. Gleirscher, B. Hummel, M. Irlbeck. On the Extent and Nature of Software Reuse in Open Source Java Projects. In *Proceedings of the 12th International Conference on Software Reuse*. Pp. 207–222. 2011.

[Kos07] R. Koschke. Survey of Research on Software Clones. In *Duplication, Redundancy, and Similarity in Software*. 2007.

[Mav] Apache Maven Project.
http://maven.apache.org/

[SMBZ07] D. Saougkos, G. Manis, K. Blekas, A. Zarras. Revisiting Java Bytecode Compression for Embedded and Mobile Computing Environments. *Software Engineering, IEEE Transactions on* 33(7):478–495, 2007.