A Reproduction of Dependency Facade: The Coupling and Conflicts between Android Framework and Its Customization

***Abstract*—This artifact aims to share the tools, scripts, and data of our research paper – "Dependency Facade: The Coupling and Conflicts between Android Framework and Its Customization", which also claims 3 badges (Functional, Reusable, and Available). To use and evaluate the artifact, users should first configure the required environments, clone the project's source code and employ the git command to specify different versions. Then users should download our provided package to get the tools and scripts, following the steps in README to reproduce the results.**

I. PAPER TITLE AND AUTHOR

We are submitting this artifact in reference to the technical paper "Dependency Facade: The Coupling and Conflicts between Android Framework and Its Customization" accepted by ICSE 2023. The authors are

1) Wuxia Jin (jinwuxia@mail.xjtu.edu.cn),
2) Yitong Dai (daiyitong0808@stu.xjtu.edu.cn),
3) Jianguo Zheng (Zz_Nut@stu.xjtu.edu.cn),
4) Yu Qu (yuq@ucr.edu),
5) Ming Fan (mingfan@mail.xjtu.edu.cn),
6) Zhenyu Huang (huangzhenyu1@hihonor.com),
7) Dezhi Huang (huangdezhi@hihonor.com),
8) Ting Liu (tingliu@mail.xjtu.edu.cn).

Besides, Yitong Dai and Jianguo Zheng are the authors that performed the reproduction.

II. INTRODUCTION

Mobile device vendors develop their customized Android OS (termed downstream) based on Google Android (termed upstream) to support new features. During the independent development, the downstream also periodically merges changes of a new release from the upstream into its development branches, keeping in sync with the upstream. Due to a large number of commits being merged, heavy code conflicts would be reported if auto-merge operations failed. Prior work has studied conflicts in this scenario. However, it is still unclear about the coupling between the downstream and the upstream (We term this coupling as the dependency facade), as well as how merge conflicts are related to this coupling.

To address this issue, we first propose the DepFCD to reveal the dependency facade from three aspects: interface-level dependencies that indicate a clear design boundary, intrusion-level dependencies which blur the boundary, and dependency constraints imposed by the upstream non-SDK restrictions. Moreover, we analyze merge conflicts on the dependency facade.

To support the study, we collect four open-source downstream projects and one industrial project, with 15 downstream and 15 corresponding upstream versions.

III. PURPOSE OF THE RESEARCH ARTIFACT

This artifact contains our tool and the data:

- DepFCD is the dependency facade identification tool we implemented, based on the methodology illustrated in our technical paper;
- the raw data we collected for experiments in our technical paper.

Our DepFCD is the first tool to reveal and model the dependency (from three aspects including interface-level dependency, intrusion-level dependency, and dependency constraints) between the Android framework and its customization.

IV. BADGES CLAIMED

We are claiming 3 badges (Functional, Reusable, and Available) under the Artifacts Evaluated and Available sections.

1. Functional. The artifact is completely documented to record an inventory of the tool, the scripts, the results, and the documentation about how to exercise the artifact. The artifact is consistent because it contains DepFCD and the intermediate data used to produce the results in the technical paper. The artifact is complete because it includes all the components relevant to the technical paper. The artifact is exercisable because the scripts to produce the results can execute successfully using the commands provided in the documentation.

2. Reproduced. We strongly believe that our DepFCD tool can be reproduced and extended. DepFCD provides a user-friendly command-line interface, prompting help and usage guidance. The source code of DepFCD is written in Java and Python, hence easy to extend its functionality. Moreover, the dataset we provided is clean and complete to facilitate continuous research in the community.

3. Available. We publish the whole data, scripts, and tools at Github repo[1] so that the community can reuse, improve, and extend it.

[1]https://github.com/xjtu-enre/DepFCD

V. TECHNOLOGY SKILLS ASSUMED BY THE REVIEWER

We assume that the reviewers or users can execute the basic commands in a shell environment, such as *git, pip3, python3, tar, unzip*.

VI. SOFTWARE REQUIREMENTS

Our DepFCD has been developed and tested on both windows and Linux environments. Due to the large size of the AOSP and its customizations, the memory heap of the machine is better to over 30G. Before using DepCD, it requires the following environment to be installed.

1) JDK 11 or higher
2) Python 3.7 or higher
3) latest version of Git
4) latest version of pip3
5) Python3 packages in our artifact documentation of *REQUIREMENTS.docx*

We introduce how to install 3)-5) in our artifact documentation of *REQUIREMENTS.docx*.

VII. HOW TO START

The artifact folder we shared is named **DepFCD_AE**. This folder includes reusable tools and scripts that we intend to share publicly. More importantly, the folder also includes the corresponding data studied in our paper.

After downloading and unpacking our **DepFCD_AE.zip** file, **method**, **scrpits**, **data**, and **document** folders can be generated. The **document** folder presents several guidelines, including *README.docx*, *REQUIREMENTS.docx*, and *INSTALL.docx*, to illustrate how to obtain, configure, install, and use the artifact in detail.

First, please refer to the *README.docx* file. This file gives details on the content of this artifact package, the mapping between the generated results and parts in the paper, and how to use the package for replicating the study results and reusing the code and data for future studies.

Then users or reviewers can go to the *INSTALL.docx* file. This document presents installation instructions on how to prepare the executable software environment, which the *REQUIREMENTS.docx* specified, and illustrates a very basic usage example to reproduce our study but covering the complete study process.

After installing the environment, the users or reviewers can execute our tool. Please note that our tool in the paper needs to fetch all commit history and retrieve the whole refactoring operations of the analyzed project, which is time-consuming. To use our tool more conveniently, you can directly use the "all_base_commits.csv" and "ref.json" for LineageOS-18.1 located in our folder "data//Methodology//The detection of Dependency facade//Entity Ownership Identification". The only thing you need to do is moving them to the *output_path* which is specified in the provided command.

ACKNOWLEDGMENT

This work was supported by the National Key R&D Program of China (2022YFB2703500), National Natural Science Foundation of China (62272377, 61902306, 62002280, 61721002, 61833015), and sponsored by the Honor Corporation, CCF-Huawei Open Research Fund, Fundamental Research Funds for the Central Universities (xzy012020109, xxj022019001, xzy012020009), CCF-AFSG Research Fund, China Postdoctoral Science Foundation (2020M683507, 2019TQ0251, 2020M673439), and the Young Talent Fund of Association for Science and Technology in Shaanxi, China.