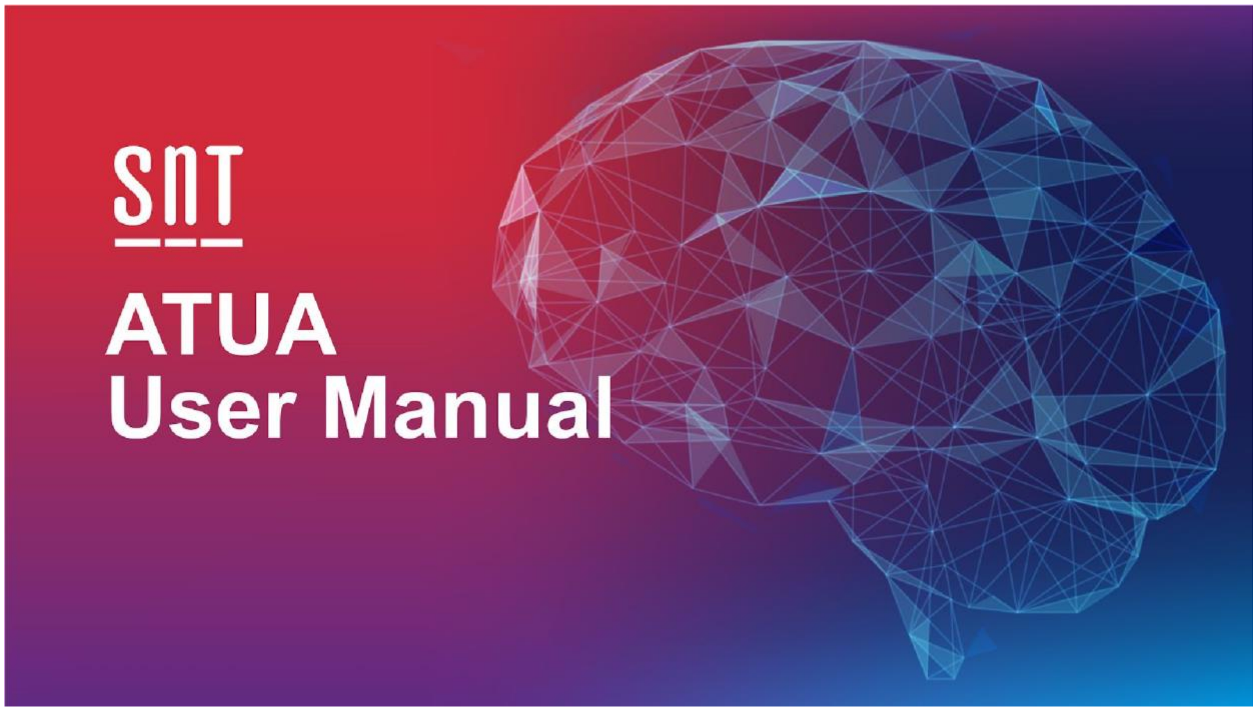

# ATUA User Manual

Prepared by Chanh Duc NGO

Version: V1.1.1

# 1 Introduction

ATUA is a test automation tool for mobile Apps. It focuses on testing methods updated in each software release. The tool aims to maximize the coverage of updated methods and their instructions with a reduced set of inputs.

The ATUA software package includes four components:

1. AppDiff

   - AppDiff is an extension of LibScout, a lightweight and effective static analysis tool to detect third-party libraries in Android/Java apps.
   - AppDiff finds code changes across two versions of an app, i.e. changes in methods with the same signature, added/deleted classes/packages.
   - AppDiff works on compiled bitcode.

2. Extended-Gator

   - Extended-Gator is an extension of Gator (Program Analysis Toolkit for Android).
   - Extended-Gator enhances Gator's flowgraph analysis and extends Gator's WTG (Window Transition Graph) construction to generate EWTGs (Extended Window Transition Graphs), which are then used as inputs for the testing process lead by ATUA

3. Extended-DM2-Instrumenter

   - Extended-DM2-Instrumenter is an extension of DroidMate-coverage (a component of DroidMate-2).
   - Extended-DM2-Instrumenter log an extended set of information with respect to the original DroidMate-coverage. Precisely, it also records the instruction's method information so that ATUA can measure the coverage of instructions as well as the one of methods.

4. ATUA

   - ATUA is an extension of DroidMate-2 (DM2), a GUI test automation framework for Android Apps.
   - The main ATUA components are the ATUA Model Feature and the ATUA Strategy, which extends the corresponding interfaces in DM2.

# 2 Requirements

- JDK 1.8+ is required for all the tools specified in the following sections.
- Python 3 is required for Extended-Gator
- Android SDK
- Android 23+ emulator or device

# 3 Installation

## 3.1 AppDiff

- Download the project from: https://github.com/SNTSVV/AppDiff
- Installation guide: https://github.com/SNTSVV/AppDiff#readme

## 3.2 Extended-Gator

- Download the project from: https://github.com/SNTSVV/Extended-Gator
- To build the project, run the following command from the project's root directory:

```
./gator b
```

## 3.3 Extended-DM2-Instrumenter

- Download the project from: https://github.com/SNTSVV/Extended-DM2-Instrumenter
- The project is gradle-based and is setup with gradle wrapper. To build the project, run the following command from the project's root directory:

```
./gradlew build
```

## 3.4 ATUA

- Download the project from: https://github.com/SNTSVV/ATUA
- The project is gradle-based and is setup with gradle wrapper. To build the project, run the following command from the project's root directory:

```
./gradlew build
```

# 4 Usage

## 4.1 Identify target methods and instructions

This step concerns the identification of the methods modified and introduced by an updated version. This is accomplished by executing AppDiff. This step is not necessary to run ATUA with the provided case studies.

To run AppDiff, use the following command:

```
    java -jar appdiff.jar -o diff -a <android_sdk_jar> -j <output_folder>
<path_to_oldVersionApk> <path_to_newVersionApk>
```

A diff file in json format is produced and put in the `<output_folder>`

## 4.2 Generate the EWTG App model

This step concerns the generation of EWTG App model using **Extended-Gator**.

Before executing the tool, put **the app under test (i.e., apk file)** and **the diff file** (generated by AppDiff, see 4.1) in the same folder (e.g., **apks** folder)**.** If the diff file is not provided, the app under test will be analysed as all the methods are updated.

To generate EWTG App model, use the following command from the tool's root folder:

```
    ./gator a -p <Apk Path> -outputFile <EWTG App Model path> -worker 4
-appPackage <app package of the App under test> --timeout 36000
```

- **worker**: the number of processors which will be used
- **appPackage**: the source code's package name. Some applications have their declared package name which does not match their source code's package name. This parameter allows gator to analyse these kinds of apps.
- **timeout**: in second. The default timeout of Gator is 1 hour. This parameter could help to deal with large apps which require long processing time.

## 4.3 Instrument the app under test

This step concerns the instrumentation of the app under test using Extended-DM2-Instrumenter.

Firstly, put the apk file of the app under test and the diff file into the folder **Extended-DM2-Instrumenter/apks.**

Run instrumentation from **Extended-DM2-Instrumenter**'s root folder with the following command .

```
    ./gradlew run
```

The output of this step is contained in the **instrumentedApks** folder, including the instrumented apk and the instrumented information.

You can specify customized configuration by changing the content of **args.txt** (in the project's root folder). For example:

```
    --useAppt2=true –packageName= bbc.mobile.news.v3
```

- **useAppt2**: some apps require appt2 to decompile and compile. This option is false by default.

- **packageName**: the source code's package name. Some applications have their declared package name which does not match their source code's package name. This parameter allows the tool to instrument these kinds of apps.
- Other configuration parameters:
  - o **apk**: the input folder containing the apk file of the app under test
  - o **outputDir**: the output folder containing the instrumented apk.

## 4.4 Run testing with ATUA

### 4.4.1 Prepare the testing configuration

1. Reuse defaultConfiguration-ATUA.property file or duplicate it for new purpose
2. Replace the configuration file in the **args.txt**
3. In the configuration file, consider the following properties:

- **Output.outputDir** : the relative path of the output folder of the tool. If the folder does not exist, the tool will create the new folder. *Attention: everytime ATUA is started, this folder will be cleaned.*

ex: Output.outputDir=./out60m/ATUA

- **Exploration.apksDir**: the relative path of the folder where the app under tests are located ex: Exploration.apksDir=./apk
- **Exploration.deviceSerialNumber**: the serial number of the device (got from "adb devices" command). *Attention: please make sure that the device is available before executing the tool.*

ex: Exploration.deviceSerialNumber=emulator-5554

- **Selectors.timeLimit**: testing time budget in minutes ex: Selectors.timeLimit=60
- **Selectors.actionLimit**: this is useful if you prefer the maximum number of actions as the overall budget ex: Selectors.actionLimit=1000
- **RegressionStrategy.budgetScale**: scaleFactor to be used for each phase. The default value should be 1.0 for 1 hour testing budget. For five hours, we suggest using 2.0.

### 4.4.2 Prepare the App under test

ATUA needs the instrumented apk of the App under test, with a json providing instrumentation information, and the EWTG App model file. In addition, users can provide manual inputs and manual intents (a template for these inputs is provided in the **template** folder).

Put the necessary files of the App under test into the folder *Exploration.apksDir*. The distribution comes with some case study Apps. The inputs files's formats are the following:

- **EWTG App model:** *[ appPackage]_[v1]_[v2]-AppModel.json*.
  - This file is generated by Extended-Gator (more details in 4.2).
- **Instrumented apk:** *[appPackage]_[v1]_[v2]-instrumented.apk*.
  - This file is generated by Extended-DM2-Instrumenter (more details in 4.3)
- **Instrumented info:** *[ appPackage]_[v1]_[v2].apk.json*.
  - This file contains the list of instructions and methods of the app under test. This file is generated with the instrumented file (more details in 4.3)
- **Manual input:** *[appPackage]_[v1]_[v2]-input.json*.
  - This file is expected to be filled with manual inputs to be used for testing an App.
- **Manual intent:** *[appPackage]_[v1]_[v2]-intent.json*.
  - This file is expected to be filled with manual inputs to be used for testing an App intents.

### 4.4.3 Execute ATUA

To execute ATUA, simply run the following command under the project folder:

```
./gradlew run
```

## 4.5 3.3.3 Inspecting results

The result is stored in the [**outputDir]/droidmate :**

- **coverage/:** contains details of coverage of each action (generated by DM2)
- **report/:** contains graph visualization of the entire execution (generated by DM2)
- logcat.log:
- **model/[appPackage]:** contains the model and the coverage related to updated methods.
  - **atua-report.txt:** contains the overall result in which you can find:
    - The number of statements (instructions)
    - The number of methods
    - The number of updated methods
    - The number of statements belong to the updated methods
    - The coverage
    - Some additional information including:
      - List of covered updated methods
      - List of uncovered update method

  - The coverage ratio after each phase
  - Unreachable windows
- **coverage.txt:** contains the list of instructions with the first covered timestamp.
- **crashlist.**txt: contains the list of crash exceptions (generated by DM2)
- **EWTG/**: contains the enhanced EWTG's part
- **DSTG/**: contains the DSTG generated by ATUA
- **States/**: contains the captured GUITrees.
- **trace<id>.csv**: contains all the inputs generated and transitions after each action.
- **Images/**: contains the screenshots of apps. ATUA takes a screenshot after every input being triggered.
- **actionCoverage.csv**: contains the summary of coverage of each action.
- **actionCoverageHTMLReport**: contains detail report of each target action (i.e. the actions increase the updated code coverage) under seperated HTML page. The name of an HTML file corresponds to the action's id that the page reports.