User Guide v1

Author	Vincenzo De Rosa
Last change	23/03/2021
Last visit to links	23/03/2021

Table of Contents

1 Data Preparation

Before running the tool, you must prepare the data to be given as input. They must be present on three files: *logs.csv*, *bugs.csv* and *metrics.csv*.

1.1 Logs

In our study, we used 2 Eclipse software, but nothing prohibits the use of any other software, as long as the input formats for the tool are respected.
From the Git repositories we cloned the software end extracted the logs.
To extract software logs, open the git console inside the cloned directory and run the following commands:

echo ""commit hash","parent hash","author","date","subject","changed files"" > logs.csv

git log master --date=local --pretty="%xb5%H%x2C%P%x2C%an%x2C%as%x2C%x22%s%x22%x2C" --numstat --no-merges | tr "\n" " " | tr "µ" "\n" >> logs.csv

Check that the *".logs"* file does not have any incorrectly formatted characters, otherwise the tool may return an error. On the other hand, if an error appears later, it will also report what line of the file it is on.
From the logs you can choose the commit hash you want and download its version, starting from the cloned one, through the following command:

git clone `URLTORepository`

cd `into your cloned folder`

git checkout `commithash`

1.2 Bugs

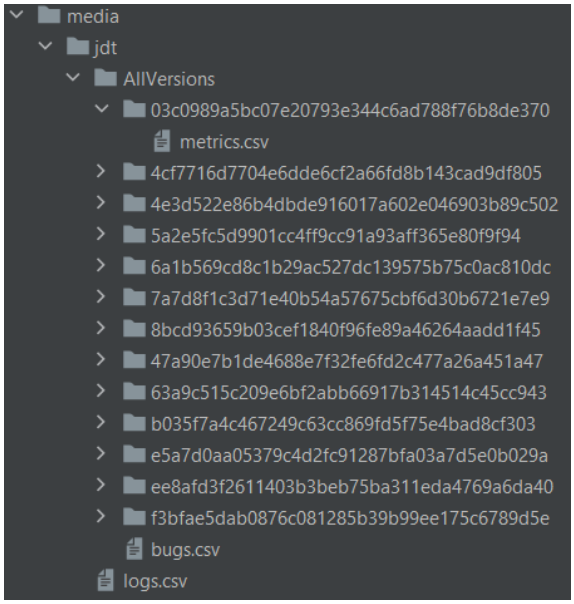
From Bugzilla we downloaded the .csv of the bugs.
In the Bugzilla search, select the desired software and select *RESOLVED*, *VERIFIED* and *CLOSED* as status and *FIXED* as resolution.
After searching, download the *.csv* file using the command provided on the page and save it as *bugs.csv*.

1.3 Metrics

Upload the downloaded versions within the Understand software and extract the metrics into a *.csv* file named *metrics.csv*.

2 Django Project

In the django project there is a folder called *"media"*, inside it we are going to place all the necessary files following the scheme below.



Note that the names given to the folders must be the same as those that will be used within the tool.

At the first level create a folder with the name of the software and insert the file *logs.csv* inside it. Also create a folder for each version (if you want to insert more than one). Inside each version insert the file *bugs.csv*. At the next level create a folder with the hash of the commit and insert inside it the file *metrics.csv*.

Please note that in the example shown there is only one version, named *AllVersions*, since a temporal criterion was chosen for the selection of versions. If you choose based on releases the structure should look like this:

media->jdt -> logs.csv
4.0 -> bugs.csv
{hash of version committed} -> metrics.csv
{hash of version committed} -> metrics.csv
4.1 -> bugs.csv
{hash of version committed} -> metrics.csv
{hash of version committed} -> metrics.csv

2.1 Software And Version Creation

```python
def create_software(request):
    # software creation
    software, created = Software.objects.get_or_create(name='swt',
                                                       log_file_path='media/swt/logs.csv',
                                                       source_root_folder='bundles')
    software.create_cleaned_log_instances()

    # version creation
    version, created = Version.objects.get_or_create(name='AllVersions',
                                                     bug_file_path='media/swt/AllVersions/bugs.csv',
                                                     software=software)
    version.create_bug_instances_from_csv()
    version.create_tracked_bugs()

    return HttpResponse('OK')
```

In *ProgettoTesi -> DataRetrival -> views.py* there is the view shown above.
Change the data to create whatever software and version you want inserting the exact path where you put the files. The field *"source_root_folder"* indicate the folders where the tools will consider for the bugs. See how is your software's structure and add your preferred folders delimiting them with *'@#'*, the tool will split parsing these chars. Examale: *'bundles@#folder1@#folder2@#...folderN'*.
This view will create a software and its logs, and, for each version will create bug instances and the tracked bugs searching a reference between the logs and the bugs. The view is executable with the URL:

http://127.0.0.1:8000/dataretrival/software/create_software/

2.2 Measures Creation

```python
def create_measures_rq1(request):
    version = Version.objects.get(name='AllVersions', software__name='jdt')
    # Detailed dataset
    version.dataset_rq1('b035f7a4c467249c63cc869fd5f75e4bad8cf303')
    version.dataset_rq1('4e3d522e86b4dbde916017a602e046903b89c502', '2015-01-03', '2015-06-30')
    version.dataset_rq1('8bcd93659b03cef1840f96fe89a46264aadd1f45', '2015-07-01', '2015-12-31')
    version.dataset_rq1('f3bfae5dab0876c081285b39b99ee175c6789d5e', '2016-01-01', '2016-06-29')
    version.dataset_rq1('e5a7d0aa05379c4d2fc91287bfa03a7d5e0b029a', '2016-06-30', '2016-12-25')
    version.dataset_rq1('47a90e7b1de4688e7f32fe6fd2c477a26a451a47', '2016-12-26', '2017-06-30')
    version.dataset_rq1('5a2e5fc5d9901cc4ff9cc91a93aff365e80f9f94', '2017-07-01', '2017-12-26')
    version.dataset_rq1('63a9c515c209e6bf2abb66917b314514c45cc943', '2017-12-27', '2018-06-30')
    version.dataset_rq1('4cf7716d7704e6dde6cf2a66fd8b143cad9df805', '2018-07-01', '2018-12-29')
    version.dataset_rq1('ee8afd3f2611403b3beb75ba311eda4769a6da40', '2018-12-30', '2019-06-28')
    version.dataset_rq1('6a1b569cd8c1b29ac527dc139575b75c0ac810dc', '2019-06-29', '2019-12-29')
    version.dataset_rq1('03c0989a5bc07e20793e344c6ad788f76b8de370', '2019-12-30', '2020-06-30')
    version.dataset_rq1('7a7d8f1c3d71e40b54a57675cbf6d30b6721e7e9', '2020-07-01', '2020-12-24')
```

In *ProgettoTesi -> Misuration -> views.py* there is the view shown above for the dataset of RQ1. In the image is reported only a piece of the full function.
There is a view for each research question and reachable with the url:

http://127.0.0.1:8000/measurement/create_measures_rq1/
http://127.0.0.1:8000/measurement/create_measures_rq2/

…

The function takes as input the commit hash from which it takes the measures file and two dates, one start and one end date.
The two dates are not mandatory and are used to narrow down the time range in which to consider bug fixes. If not selected, **no information** about bugs and measures will be collected. This case is only useful if you want a snapshot of the software from which to start comparing it with later versions.
In our case, we selected only two versions with a wide data range for the general analysis. While, for the detailed analysis, multiple versions with multiple narrow date ranges.
After the views are executed (it will take a long time depending on the number of logs and bugs) the datasets in *.csv* format will appear within the project directory.
From the url

http://127.0.0.1:8000/dataretrival/software/list/,

instead, it is possible to navigate inside the different pages for the visualization of the data.