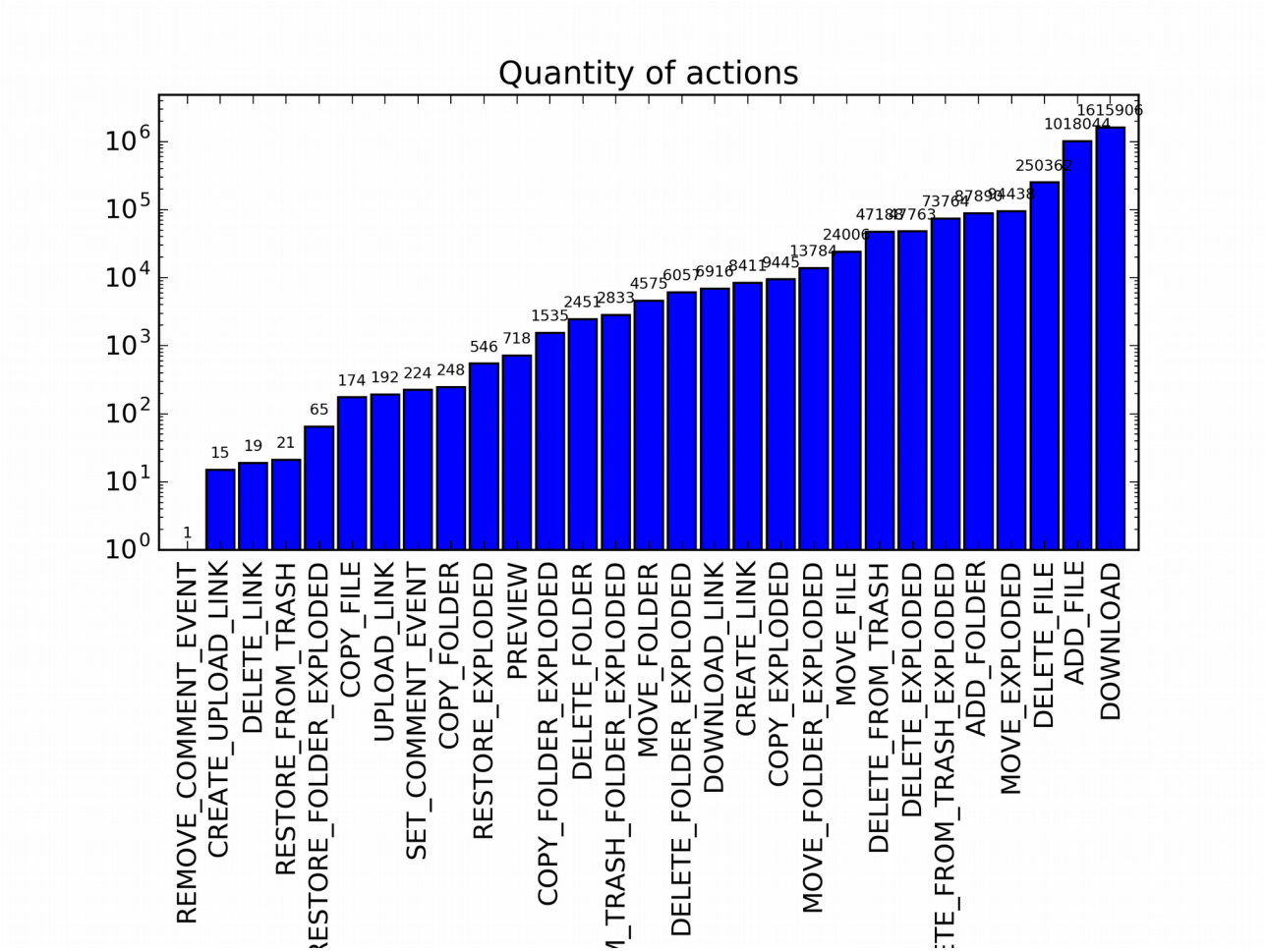
Kamil Pluciński

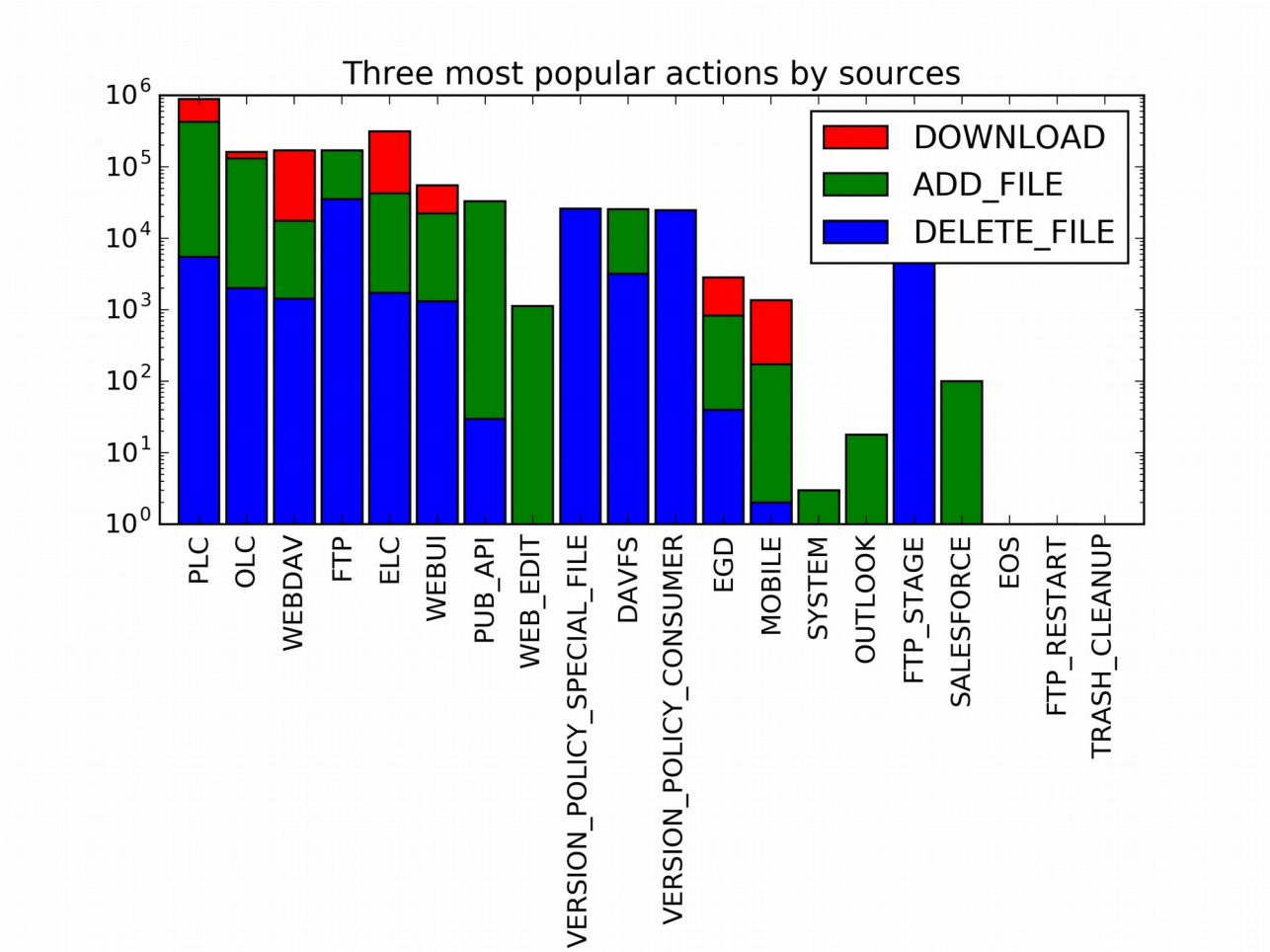
Description of the solution

The datalog set is large so I thought that I should read all files only once and import information line by line. I knew the dataset could grow and I would have to create more charts. A data class shouldn't know anything about charts. It only shares information. I decided to use design pattern - observer. If I wanted to add a new chart I wouldn't have to change the data class. This solution is consistent with SRP and OCP. For all exercises I used python 3.5 with libraries: unittest, matplotlib, json and a little bit of bash scripts with excel.

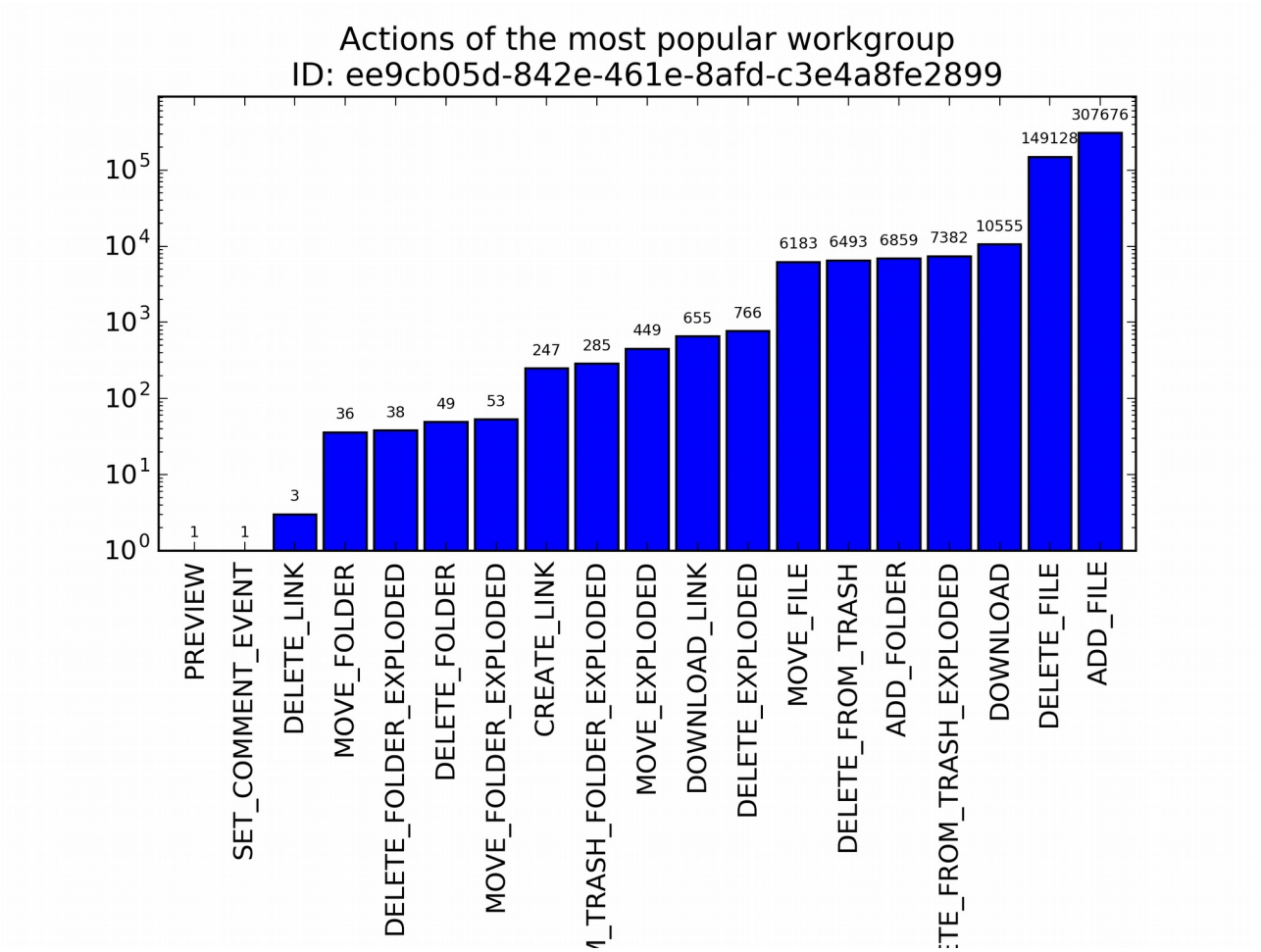
Statistic



By analyzing the data showed on the chart, I come to the conclusion that the most popular action is downloading. Adding files has almost the same popularity as downloading, and so does deleting. On the chart I can see that operations on links have the lowest popularity.



The bar chart is about sources of the three most popular actions. For download and add_file actions the largest source is PLC, although for delete_file action it is the FTP_STAGE. Sources: EOS, FTP_RESTART and TRASH_CLEANUP weren't used.



This chart shows actions of the most active workgroup. On the first place there isn't DOWNLOAD but ADD_FILE. On the second place there is DELETE_FILE and DOWNLOAD comes after that. Actions from DOWNLOAD to MOVE_FILE have almost the same popularity. After that, there is a sudden decrease in popularity and the values go down slightly.

Tests

All users

Number	Action
1	REMOVE_COMMENT_EVENT
15	CREATE_UPLOAD_LINK
19	DELETE_LINK
21	RESTORE_FROM_TRASH
65	RESTORE_FOLDER_EXPLODED
174	COPY_FILE
192	UPLOAD_LINK
224	SET_COMMENT_EVENT
248	COPY_FOLDER
546	RESTORE_EXPLODED
718	PREVIEW
1535	COPY_FOLDER_EXPLODED
2451	DELETE_FOLDER
2833	DELETE_FROM_TRASH_FOLDER_EXPLODED
4575	MOVE_FOLDER
6057	DELETE_FOLDER_EXPLODED
6916	DOWNLOAD_LINK
8411	CREATE_LINK
9445	COPY_EXPLODED
13784	MOVE_FOLDER_EXPLODED
24006	MOVE_FILE
47188	DELETE_FROM_TRASH
47763	DELETE_EXPLODED
73764	DELETE_FROM_TRASH_EXPLODED
87890	ADD_FOLDER
94438	MOVE_EXPLODED
250362	DELETE_FILE
1018044	ADD_FILE
1615906	DOWNLOAD
3317591	**7,55**

Users without a useragent

Number	Action
1	RESTORE_FROM_TRASH
4	DELETE_FOLDER_EXPLODED
18	MOVE_FOLDER
59	RESTORE_FOLDER_EXPLODED
75	DELETE_FOLDER
211	MOVE_FOLDER_EXPLODED
441	RESTORE_EXPLODED
539	DELETE_EXPLODED
1388	MOVE_EXPLODED
2833	DELETE_FROM_TRASH_FOLDER_EXPLODED
6500	MOVE_FILE
7090	ADD_FOLDER
47085	DELETE_FROM_TRASH
73644	DELETE_FROM_TRASH_EXPLODED
151761	ADD_FILE
187489	DOWNLOAD
197580	DELETE_FILE
676718	**29,20**

I created three tests for checking if the data is correct. The first test check if files exist. The second test check if all json objects contain all required fields. In the last test I check values in all fields, mainly if a field contains value in the right type and for long strings with separators I check a number of parts. The dataset passed all of my tests.

Discrepancies

I found a lot of logs with the value of ‘None’ in the userAgent field. On wikipedia it is written that server applications shouldn't depend on a userAgent field because information in this field is easy to delete or modify. In addition, action ‘DELETE_FILE’ for an average user represent only 7.5% of all actions, but for users with ‘None’ in userAgent field this is 29% and it is most popular action.
Values was calculated using bash and excel.

Bibliography

https://en.wikipedia.org/
https://stackoverflow.com/