much earlier, it took engineers a lot of time to confirm it. In this case, if we take the last change time of bug, majority of the bug's fix time would be inaccurate. Hence, we defined fix time as the time between the creation time and the change of the bug status as *RESOLVED*. We define this as *bug fix time*.

After the calculation of the fix time in hours, we converted them into four classes: *very slow*, *slow*, *fast*, and *very fast*. The categorization has been done based on interquartile ranges of bug fix data, as it was previously proposed in [1]. Table II shows the number of bugs that fall into each category. Also the number of the bugs in these ranges are uniformly distributed. Table III shows the statistics about fix time.

TABLE I. AVERAGE TIME BETWEEN STATUS CHANGES

Status history	*Average hour*	*Number of the bugs*
Resolved to Verified	1772.30	2229
Resolved to Closed	<0.01	1
Verified to Closed	0	0

TABLE II. FIX TIME CLASSES AND THEIR RANGES

Classification	*Range*	*Number of the bugs*	*Percentage*
Very fast	1.0 - 47.0	3819	0 - 25%
Fast	47.0 - 169.0	3933	25 - 50%
Slow	169.0 - 673.0	3877	50 - 75%
Very Slow	673.0 - 90829.0	3884	75 - 100%

TABLE III. FIX TIME DESCRIPTIVE STATISTICS

Metric name	*Bug fix time*
Total instance count	15513
Mean	1581.55
Standard deviance	5357.76
Minimum	1.0
Maximum	90829.0
Quartile 25	47.0
Quartile 50	169.0
Quartile 75	673.0

D. Features

Our final dataset includes 15513 bugs reported for JavaScript Engine component and 20 attributes associated with these bugs in order to predict bug fix times. We provide an explanation for each of the bug attributes below.

- assigned to, the person that is responsible to fix the bug
- number of the keywords, keywords can be used to categorize bugs, increasing number of the keywords may show the its dependency.
- number of the cc, when any changes occurs within the bug, these persons get notification. The number of the cc list may show the importance of the bug.
- comment count, number of the comments may show the importance of the bug.
- the creation month of the bug, the creation day of the bug, the creation hour of the bug}, the temporal data may contain lots of effects such as, the developer, reporter may be off day, the time the bug created may be the last hours of the day, etc.
- the creator of the bug, different creators, reporters can have different influences, behaviors on the developers
- number of the dependent bug count, this is the number of the bug, this bug prevents fixing of other bugs
- flags number, flags can contain useful information, can indicate bug's status
- blocks number, this is the number of the bug, this bug blocks
- operating system, this is the operating system that bug is found
- priority, higher prioritizing of the bug may lead to less fix time
- platform, fix time may be affected from the platform
- severity, this shows the severe of the bug, we expect that high severity bug should have high fix times
- target milestone, the future version this bug should be fixed, for some versions the bugs should be resolved, thus the version may speed up the fix of it
- version, it shows the version number that this bug affects too
- votes, people can participate by voting a related bug, and it shows interest of the persons
- summary, summary of the bug
- description, description of the bug

Besides this, in the bug description field, we see that some bug's description fields have the same text blocks, and this inevitably affects text mining algorithms in a negative way. For example, the bug number *476* has sequences of words such as *"Created by Torsten Rüger (torsten@ponton-hamburg.de) on Monday, July 13, 1998 4:56:09 AM PDT Additional Details*", "*Updated by Mike McCabe (mccabe@netscape.com) on Monday, July 13, 1998 1:18:59 PM PDT Additional Details* ". We decided to remove these repeated texts from the description field. In our filtered set, we had only one bug that has this type of word sequences, and we removed that bug from our analysis.

IV. METHODOLOGY

A. Design

In this study, we developed three models. The first model (Model I) uses bug attributes only, the second model uses only textual descriptions of bugs only (Model II), whereas the third model combines the two (Model III): It initially analyzes bugs with respect to the similarity of their bug descriptions, and selects a subset of bugs. Then it uses bug attributes of these selected bugs and predicts the fix time. For the third model, it is necessary to have sufficient number of bugs after the first