GitHub Analysis: Detecting Bad Smells and Early Smoke in Repository

Smit Anand
Department of Computer Science
North Carolina State University
Raleigh, USA
Email: sanand2@ncsu.edu

Mahnaz Behroozi
Department of Computer Science
North Carolina State University
Raleigh, USA
Email: mbehroo@ncsu.edu

Erick Draayer
Department of Computer Science
North Carolina State University
Raleigh, USA
Email: ecdraaye@ncsu.edu

Shama Yazdani
Department of Computer Science
North Carolina State University
Raleigh, USA
Email: syazdan@ncsu.edu

***Abstract*—Bad smell detection in software development is crucial especially when it comes to software development teamwork. Each bad smell is a red flag which suggests that in the near future the team will experience problems as the project proceeds. As a result, one of the main concerns of software engineering is to define bad smells in teamwork and how to detect them in their early stages of forming.**

Keywords—component; Bad code smell, software engineering, teamwork

I. Introduction

This report evaluates three repositories for Software Engineering Spring 2017 term project. The data collected here represents the amount of work done by team members in the form of number of commits, number of issues closed and opened by each team member and such other queries. We have graphs representing variations in the project. The purpose of this analysis is to detect and show signs of bad smells through real world examples and not to condemn the teams themselves. We have also listed some features which could help determine future bad smells in a project. We believe that monitoring these features will help future teams to easily identify what they are doing wrong and how they can correct it and possibly prevent bad smells in their project.

II. Data Description

A. Data collection method

We used gitable.py code for extracting data from GitHub repositories. Few changes were made to the code so that the issues without labels could also be grabbed as we noticed that substantial amount of data was being lost due to removal of issues without label. We also changed the code so as to grab the progress of team by their weekly commits per user data. Then the code was modified to introduce anonymity. A separate mapping file was created which contained the mapping of actual team id with anonymized team Id. The users within the team were also anonymized.

B. Data format

The files extracted were in a .csv format and contained the structure described below.

1) *Table 1- Weekly Commit data:*

Columns
Week_no
Commits
Additions
Additions_per_commit

2) *Table 2- Individual commit data:*

Columns
Commit_id
Commit_user
Commit_time
Commit_message

3) *Table 3- Milestone data:*

Columns
Milestone_id
Milestone_title
Created_at
Due_at
closed_at

4) *Table 4- Issues data:*

Columns
Issue_id
When
Action
What
User
Milestone

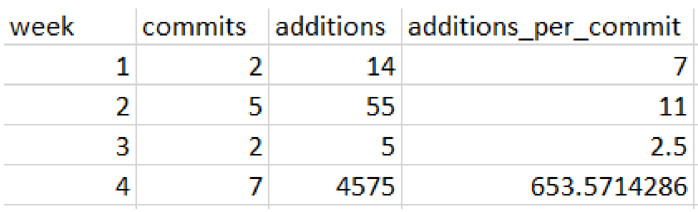
5) *Table 5- Comments data:*

Columns
Comment_id
Issue_id
User_id
Created_at
updated_at

C. *Data Sample*

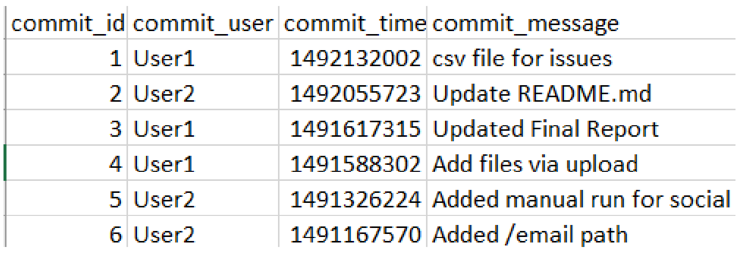
The following are data samples of the above-mentioned tables:

6) *Table 1- Weekly Commit data*: Following is sample data from Weekly commit table:-



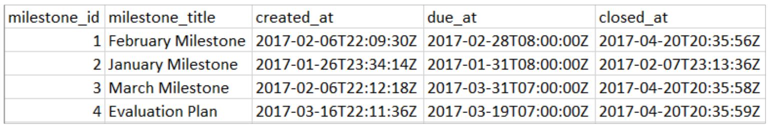
week	commits	additions	additions_per_commit
1	2	14	7
2	5	55	11
3	2	5	2.5
4	7	4575	653.5714286

7) *Table 2- Individual commit data: the* following is sample data from Weekly commit table:-



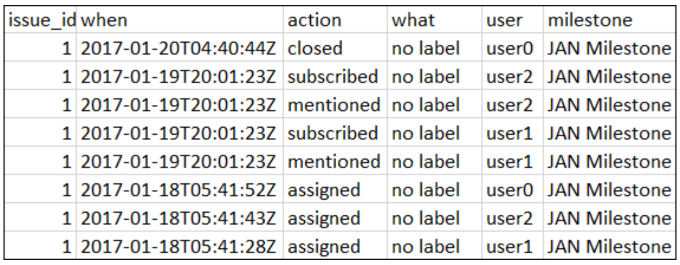
commit_id	commit_user	commit_time	commit_message
1	User1	1492132002	csv file for issues
2	User2	1492055723	Update README.md
3	User1	1491617315	Updated Final Report
4	User1	1491588302	Add files via upload
5	User2	1491326224	Added manual run for social
6	User2	1491167570	Added /email path

8) *Table 3- Milestone data*: The following is sample data from Milestone table:-



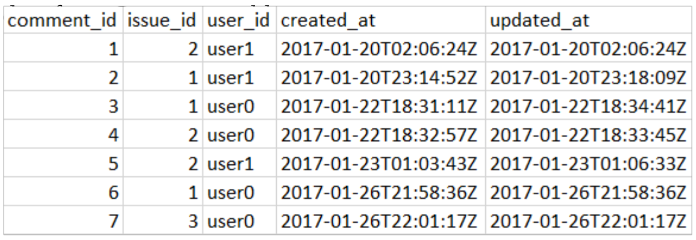
milestone_id	milestone_title	created_at	due_at	closed_at
1	February Milestone	2017-02-06T22:09:30Z	2017-02-28T08:00:00Z	2017-04-20T20:35:56Z
2	January Milestone	2017-01-26T23:34:14Z	2017-01-31T08:00:00Z	2017-02-07T23:13:36Z
3	March Milestone	2017-02-06T22:12:18Z	2017-03-31T07:00:00Z	2017-04-20T20:35:58Z
4	Evaluation Plan	2017-03-16T22:11:36Z	2017-03-19T07:00:00Z	2017-04-20T20:35:59Z

9) *Table 4- Issues data*: The following is sample data from Issue table:-



issue_id	when	action	what	user	milestone
1	2017-01-20T04:40:44Z	closed	no label	user0	JAN Milestone
1	2017-01-19T20:01:23Z	subscribed	no label	user2	JAN Milestone
1	2017-01-19T20:01:23Z	mentioned	no label	user2	JAN Milestone
1	2017-01-19T20:01:23Z	subscribed	no label	user1	JAN Milestone
1	2017-01-19T20:01:23Z	mentioned	no label	user1	JAN Milestone
1	2017-01-18T05:41:52Z	assigned	no label	user0	JAN Milestone
1	2017-01-18T05:41:43Z	assigned	no label	user2	JAN Milestone
1	2017-01-18T05:41:28Z	assigned	no label	user1	JAN Milestone

10) *Table 5- Comments data: The* following is sample



comment_id	issue_id	user_id	created_at	updated_at
1	2	user1	2017-01-20T02:06:24Z	2017-01-20T02:06:24Z
2	1	user1	2017-01-20T23:14:52Z	2017-01-20T23:18:09Z
3	1	user0	2017-01-22T18:31:11Z	2017-01-22T18:34:41Z
4	2	user0	2017-01-22T18:32:57Z	2017-01-22T18:33:45Z
5	2	user1	2017-01-23T01:03:43Z	2017-01-23T01:06:33Z
6	1	user0	2017-01-26T21:58:36Z	2017-01-26T21:58:36Z
7	3	user0	2017-01-26T22:01:17Z	2017-01-26T22:01:17Z

III. BAD SMELLS

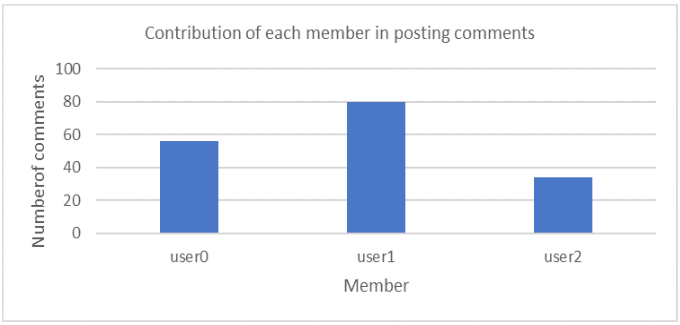
We detected the following bad smells in the analysis of 3 teams.

D. *Number of comments per user*

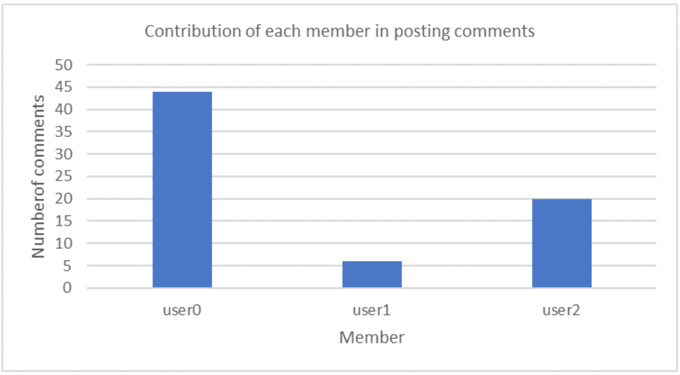
11) *Feature Detection-*
For detecting this feature we considered the number of all comments made by each user.

12) *Feature Detection Results:*

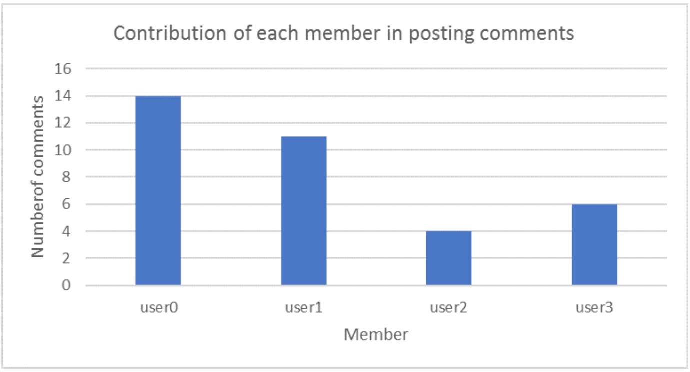
For Team1:



For Team2:



For Team 3:



Each bar in these bar charts shows the total number of comments each user submitted on issues throughout the project. The first conclusion from these results is that the more equal the bars are to each other the less likely there is a red flag of bad smell of poor communication or an absent team member.

13) *Bad smell detector*

This feature might be a good indicator of the bad smell but it can only be used towards the end of a project's lifespan. As a result, it cannot be considered for detecting smoke but for finding a forest fire.

14) *Bad Smell Results*

We can consider the mean percentage of participation of users in posting comments and calculate the standard deviation of each user from the mean percentage. The more the deviation is negative from the mean, the more idleness of the user and the more positive deviation from the mean the more labour! Another factor is standard deviation from the mean, which shows how extreme users are affecting the mean. The higher the standard deviation the stronger evidence of bad smell. The following are the results obtained from the three teams:

Team 1 with mean percentage of 33.33% and standard deviation of 13.53%:

User ID	**Percentage of participation (%)**	**Deviation from mean participation percentage (%)**	**Positive or negative deviation more than the std (%)**
User0	33	0.33	0
User1	47	14	+0.47
User2	20	13	-0.53

Team 2 with mean percentage of 23.33% and standard deviation of 19.22%:

User ID	**Percentage of participation (%)**	**Deviation from mean participation percentage (%)**	**Positive or negative deviation more than the std (%)**
User0	62.86	39.5	+20.28
User1	8.57	14.76	0
User2	28.57	5.24	0

Team 3 with mean percentage of 25% and standard deviation of 13.07%:

User ID	**Percentage of participation (%)**	**Deviation from mean participation percentage (%)**	**Positive on negative deviation more than the std (%)**
User0	40	15	+1.93
User1	31.43	6.43	0
User2	11.43	13.57	-0.5
User3	17.14	7.86	0

From the results, we can infer that in team 1 there is not much of a bad smell occurring since their standard deviation is not very notable and there is no large positive or negative deviation per user. But for team 2 there is a large standard deviation and user0 shows a notable positive deviation which means that the majority of communication was done just by this user. This is a sign of poor communication, specifically a dictator team member. Although User1 and User0 shows no deviation but that is due to the large standard deviation of this team. So while looking at deviation of each user, we have to pay attention to the standard deviation of the team. As a result, this team shows a bad smell, at least in this feature (number of comments). Team3 again, similar to Team1, has a tolerable standard deviation and the deviation of each user is not notable. So, safely we can infer that there was no bad smell, although there are comparatively more active and less active members. We can conclude from this feature that when analyzing data in larger scales (hundreds of teams) we can define a threshold which controls our decision about occurrence of bad smell.

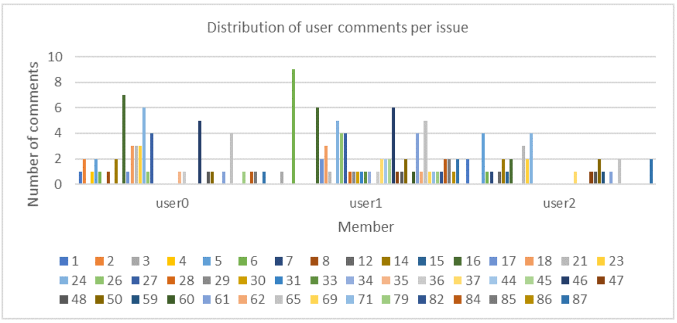
E. Number of comments per issue from each user

15) Feature Detection-

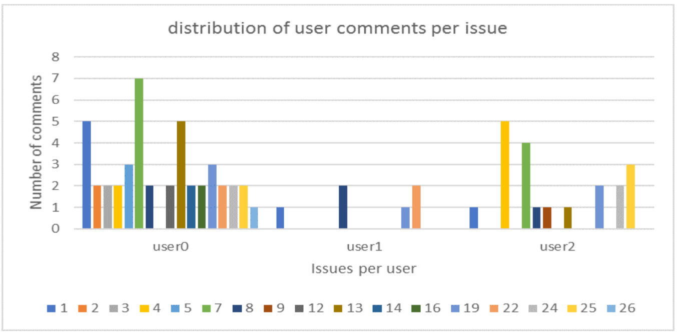
From this feature we try to understand how many comments each user has posted on each issue.

16) Feature Detection Results:

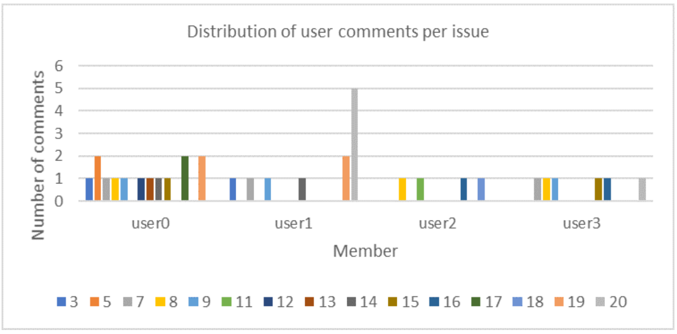
For Team1:



For Team2:



For Team3:



17) Bad Smell Detector:

For detecting bad smells from this feature, we are trying to see more signs of poor communication or absent group members. One can suggest that in the case of a bad smell, the density of a user in commenting may have a peak or there is a lack of even distribution of commenting on each issue among the users. But the point is that we cannot jump into conclusion by seeing unevenness in the number of comments in this case. Because, good fences makes good neighbors! There may be a strategy of assigning issues to each group member in a way that a member has to contribute more in the form of comments on a particular issue. In addition, the number of issues are not analogous for teams. Hence, both peaking on a particular issue and unevenness of number of comments per issue might happen in the absence of bad smell.

18) Bad Smell Results:

By making this assumption that in our SE project, all of the team members had to contribute evenly on each issue, which was not really the case, we can infer that all the above teams show an extent of a poor communication bad smell. But among the three, team 1 depicts weaker evidence of this bad smell. We can conclude from this feature that it is not always applicable. Hence, we may want to ignore it in the rest of our analysis toward bad smell detection.

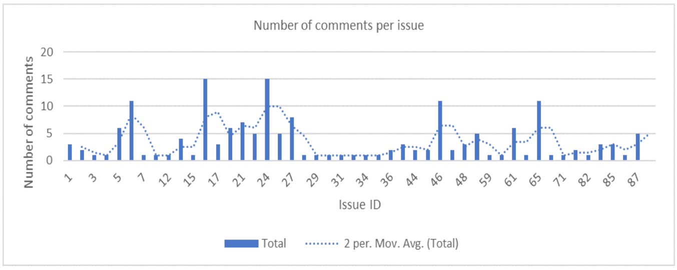
F. Number of comments per issue

19) Feature Detection-

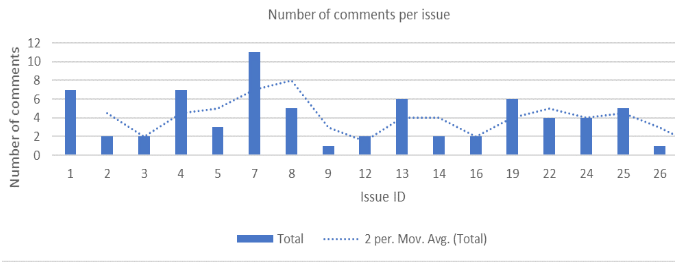
From this feature we try to understand how many comments each user has posted on each issue.

20) Feature Detection Results:

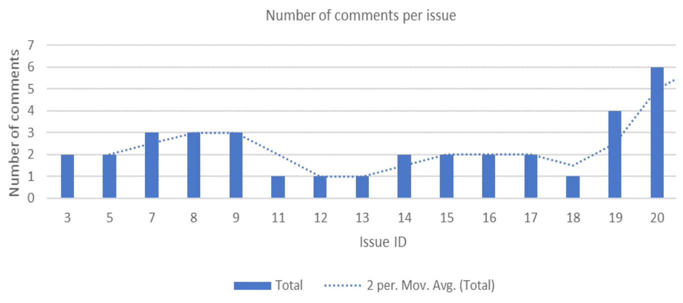
For Team1:



For Team2:



For Team3:



21) Bad Smell Detector:

For detecting bad smell from this feature, one can suggest that in the case of bad smell, the density of a user in commenting may have a peak or there is a lack of even distribution of commenting on each issue among the users. But the point is that we cannot jump into conclusion by seeing unevenness in the number of comments in this case. Because, as the good fences makes good neighbors! there may be a strategy of assigning issues to each group member in a way that a member has to contribute more in the form of comments on a particular issue. In addition, the number of issues are not analogous for teams. Hence, both peaking on a particular issue and unevenness of number of comments per issue might happen in the absence of bad smell!

22) Bad Smell Results:

By making this assumption that in our SE project, all the team members had to contribute evenly on each issue, which was not really the case, we can infer that all the above teams show an extent of bad smell. But among the three, team 1 depicts weaker evidence of bad smell. We can conclude from this feature that it is not always applicable. Hence, we may want to ignore it in the rest of our analysis toward bad smell detection.

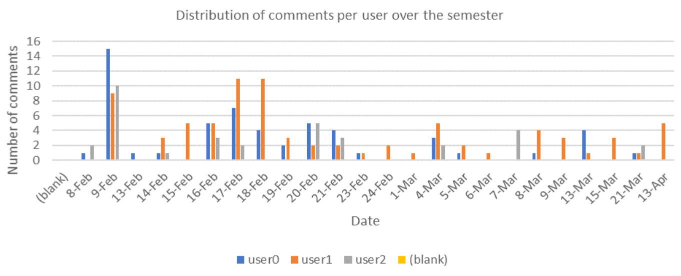
G. *Distribution of comments per user throughout the semester*

23) Feature Detection:

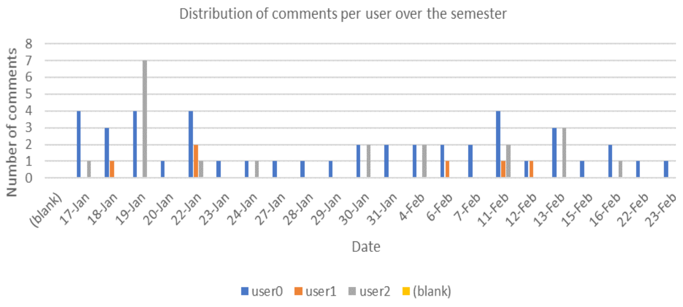
To make our previous feature a little bit more realistic and usable (at least for analysis of SE project), we decided to monitor the number of comments made over the time. This will resolve the issue of good fences makes good neighbors! Because each team member was supposed to contribute evenly in each phase of the project. Hence, if we look at the activities of each team member in a predefined time window, we can figure out whether bad smell is happening or not. From this feature we try to understand how many comments each user has posted on each issue.

24) Feature Detection Results:

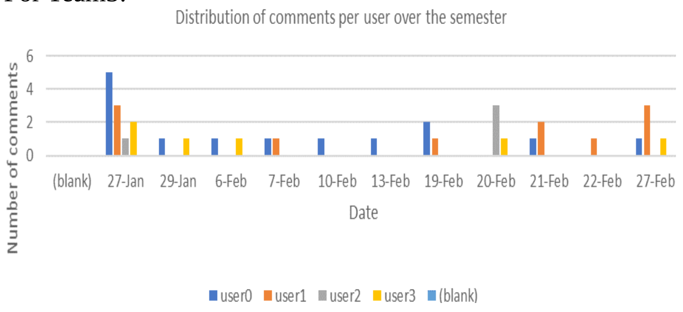
For Team1:



For Team2:



For Team3:



25) Bad Smell Detector:

For detecting bad smell we split the results into periods of each two weeks and investigated the participation of team members in those periods. In each period we calculated the mean percentage of participation and the standard deviation and then looked into each member's participation percentage and compare them to standard deviation to see if they fall out of one standard deviation from the mean.

26) Bad Smell Results:

The result obtained from our defined strategy is as follows:
For team 1 the details of biweekly activity is:

Period	**User0 (No of comments)**	**User1 (No. of comments)**	**User2 (No.of comments)**	**Mean**	**Standard deviation**
First 2 weeks	45	51	26	40.67	13.05
Second	5	12	6	7.67	3.79

2 weeks					
Third 2 weeks	6	12	2	6.67	5.03
Fourth 2 weeks	0	5	0	1.67	2.89

For team 2 the details of biweekly activity is:

Period	**User0 (No of comments)**	**User1 (No. of comments)**	**User2 (No. of comments)**	**Mean**	**Standard deviation**
First 2 weeks	25	3	12	13.33	11.06
Second 2 weeks	14	3	7	8	5.57
Third 2 weeks	5	0	1	2	2.65

For team 3 the details of biweekly activity is:

Period	**User0 (Number of comments)**	**User1 (Number of comments)**	**User2 (Number of comments)**	**User3 (Number of comments)**	**Mean**	**Standard deviation**
First 2 weeks	9	4	1	4	4.5	3.32
Second 2 weeks	5	3	3	1	3	1.63
Third 2 weeks	1	3	0	1	1.25	1.26

As results show, team 1 has been more active in comparison to other two groups. The following tables show the contribution investigation of each team member deviation from mean and to decide if there is an evidence of bad smell or not. In the tables under the last column, '+' means some bad smell found but not very sever, '++' means there is an obvious trace of bad smell and '-' means there is no bad smell or negligible amount of unevenness of activity among the team members. Evidences of bad smell occurrence: 1) High std deviation 2) evidently few or many number of comments sent by a team member which affects std. It is worth noting that just laying a member number of comments in one standard deviation from the mean does not imply that there is no bad smell, because the same member may affected the standard deviation! So for detecting bad smell in this context, first we should pay attention to distribution of the comments made by team members and then based on the mean and standard deviation decide whether there is a bad smell or not. Again, after analyzing a large dataset we can define a threshold for deciding upon the degree of deviation from std which shows bad smell. In the following tables the smelly parts are shown in boldface type.

For Team1:

Period	**Deviation of User0 from mean / more than one std?**	**Deviation of User1 from mean/more than one std?**	**Deviation of User2 from mean/more than one std?**	**Any trace of bad smell?**
First 2 weeks	+4.33/No	+10.33/No	**-14.67/Yes (-1.62)**	+
Second 2 weeks	-2.67/No	+4.33/**Yes (-0.54)**	-1.67/No	-
Third 2 weeks	-0.67/No	+5.33/**Yes (+0.3)**	-4.67/**Yes (-3.33)**	+
Fourth 2 weeks	-1.67/No	**+3.33/Yes (+0.44)**	-1.67/No	+

For team 2:

Period	**Deviation of User0 from mean / more than one std?**	**Deviation of User1 from mean/more than one std?**	**Deviation of User2 from mean/more than one std?**	**Any trace of bad smell?**
First 2 weeks	**+11.67/Yes (+0.61)**	**-10.33**/No	+1.33/No	++
Second 2 weeks	**+6/Yes (+0.43)**	**-5**/No	-1/No	++
Third 2 weeks	+3/**Yes (+0.35)**	**-2**/No	-1/No	+

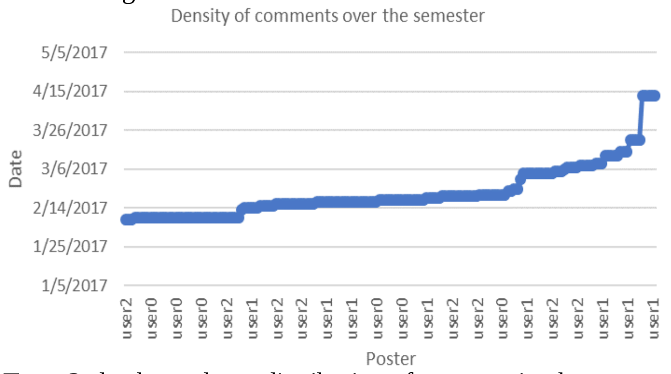
For team 3:

Period	**Deviation of User0 from mean / more than one std?**	**Deviation of User1 from mean/more than one std?**	**Deviation of User2 from mean/more than one std?**	**Deviation of User3 from mean / more than one std?**	**Any trace of bad smell?**
First 2 weeks	+4.5/**Yes (+1.18)**	-0.5/No	-3.5/**Yes (-0.18)**	-.05/No	++
Second 2 weeks	+2/**Yes (+0.37)**	0/No	0/No	-2/**Yes (-0.37)**	+

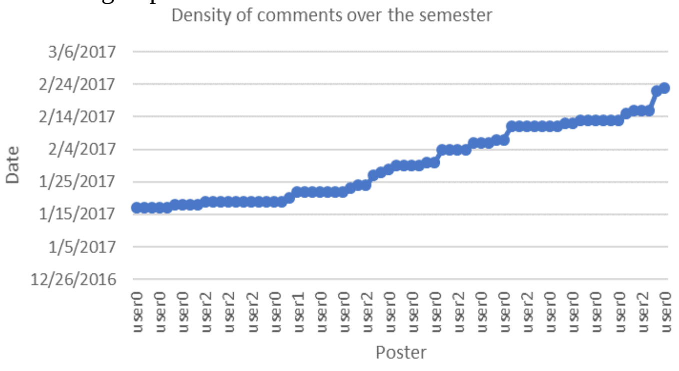
Third 2 weeks	-0.25/No	+1.75/**Yes (+0.49)**	-1.25/No	-0.25/No	-

Results show that teams' activity are not comparable enough since obviously the amount of participation in team 3 is not as much as team 1, although, there is not noticeable bad smell in team 3. One possibility is that team members were communicating outside github. So we cannot strictly infer bad smell from analyzing comments. But this feature is capable of finding smoke in its early stage since it monitors team activity in different time windows and one can change the time window size based on how strict s/he want to analyze bad smell. In order to illustrate comment distribution throughout the semester we tried another form of visualization which shows the density of comments chronologically.

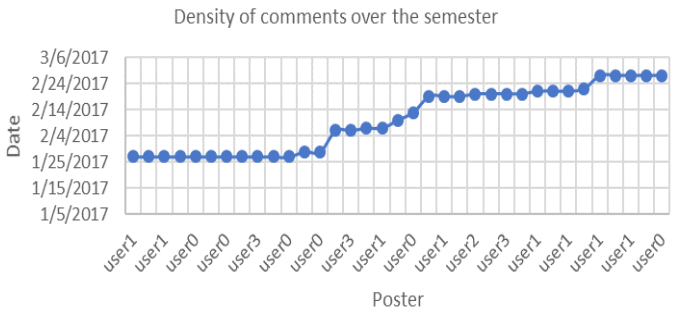
For team 1 we found that the amount of participation in commenting was dense:



Team 2 also has a dense distribution of commenting but not as dense as group 3:

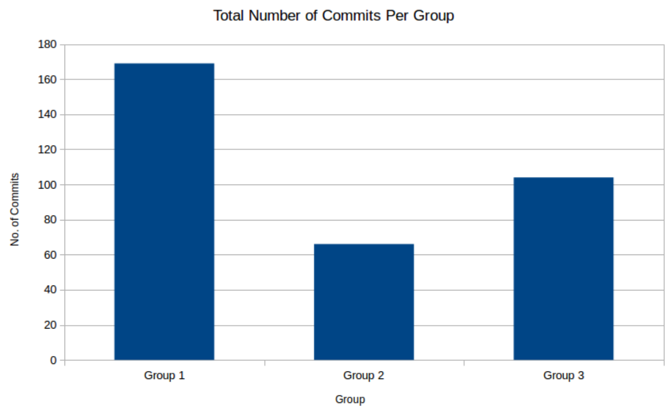


Team 3 has the sparsest contribution in comparison to other two teams:



H. Commits per group

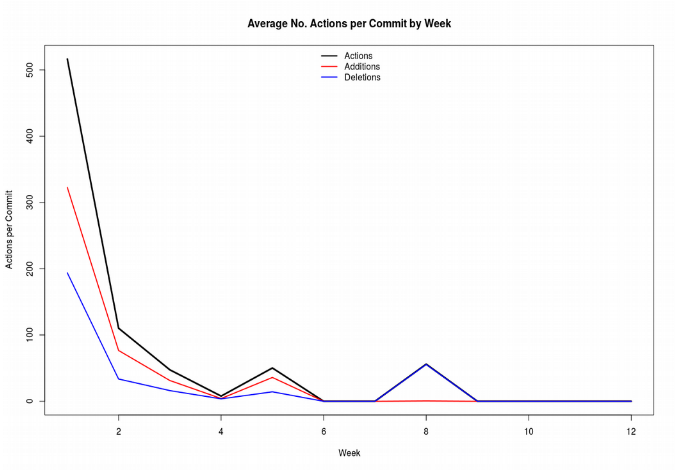
From Figure below we can see the total number of commits for the entire project from each group. Group 1 had the highest number of commits at a total of 169. The number of commits for Group 3 was a bit lower at 104. Lastly, Group 2 had the lowest amount of commits at just 66 commits. The graph shows us how much data we have available to do our bad smell detection in terms of commit information from the groups.



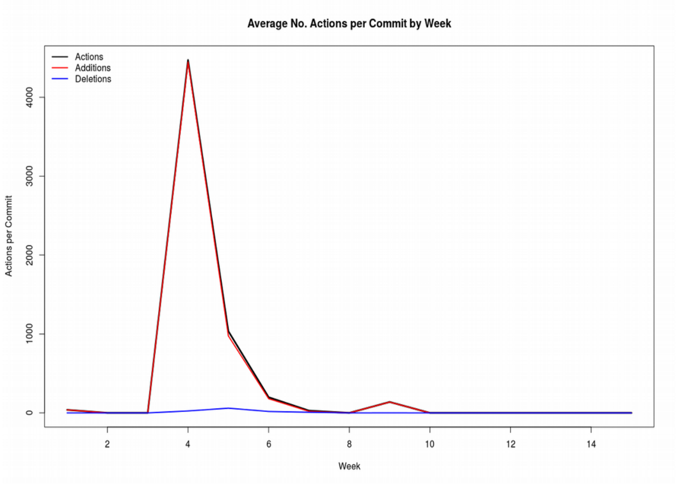
I. Average Number of Actions per Commit by Week

We define an action as a addition and/or a deletion made from a commit to the repository. The Figures 1,2, and 3 display the average actions per commit over a week long period. The red line represents additions made to the repository and the blue line represents deletions made to the repository. The two combined should equal the black line that represents the average additions and deletions per commit for that week. Each graph is over a 15 week period except group 1 because their repository wasn't created until later. Ideally a group should be pushing consistent small additions to their repository with little to no peaks. Groups should also avoid large deletions as this shows code they pushed was not effective. Also sections where the number of deletions exceeds the number of additions suggest the project isn't progressing well. All three groups exhibit a large spike in additions per commit at one point or another. This large spike occurs near the beginning of the project for groups 1 and 2, whereas for group 3 the spike happens around the March deadline. It is important to note that actions per commit are usually very low (between 5 to 70) for all of the weeks where the large spikes occur. Group 3 had the largest spike at around 20,000 additions per commit, Group 2 had the second highest at approximately 4,500 additions per commit and group 1 had the lowest which was about 500 actions per commit.

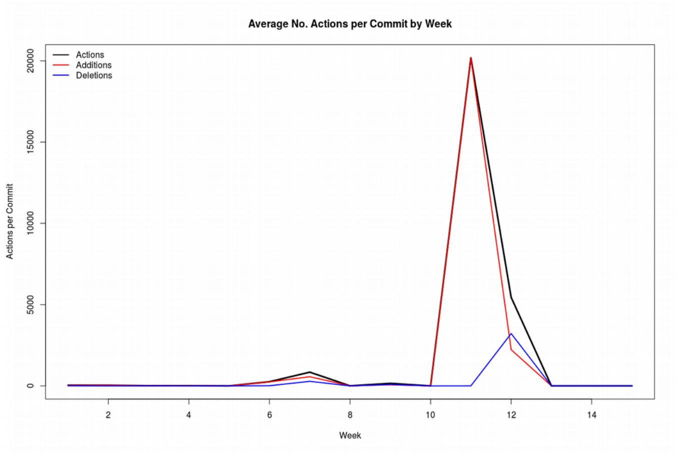
For Team 1:



For Team 2:



For team 3:

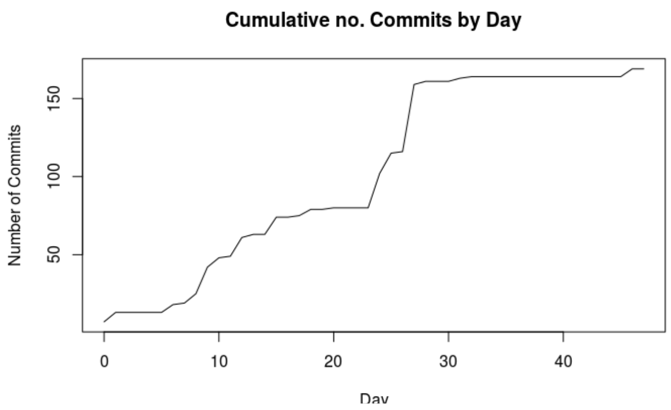


J. Commits Per Day for each group

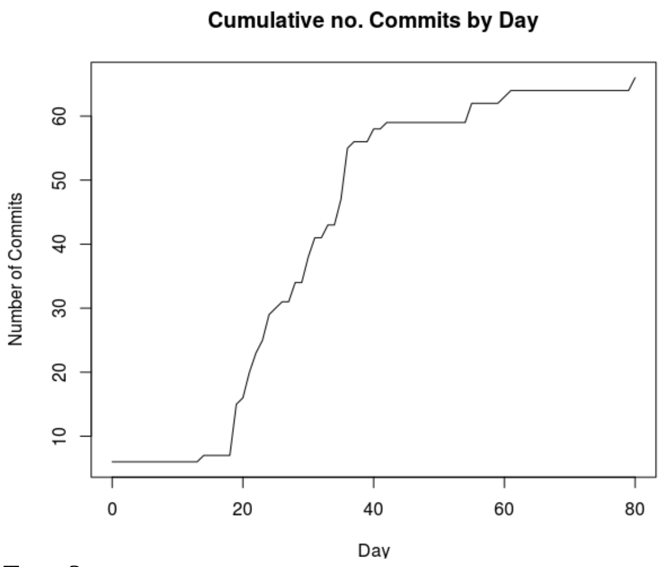
Graphs for Teams 1, 2, and 3 show the cumulative commits per day from each group. A perfect scenario would be that the groups committed a little every day, and so cumulative number of commits by day would be a straight line. But all groups were composed of humans and humans aren't perfect. All three groups have segments with little to no activity and segments with a lot of activity. The segments with a lot of activity correspond with deadlines. For example, there is a spike in activity for all three groups at around 30 day mark, which corresponds with the end of February demo. Group 3 is the only group with three spikes, with two spikes corresponding to the the beginning of the project and one corresponding to the end of March deadline. To see the linearity of each graph we calculated the R2 value of each line. The R2 values for each group are the following:

Linearity of Cumulative Number of Commits per Day	
Team	**R^2 Value**
Group 1	0.958
Group 2	0.929
Group 3	0.948

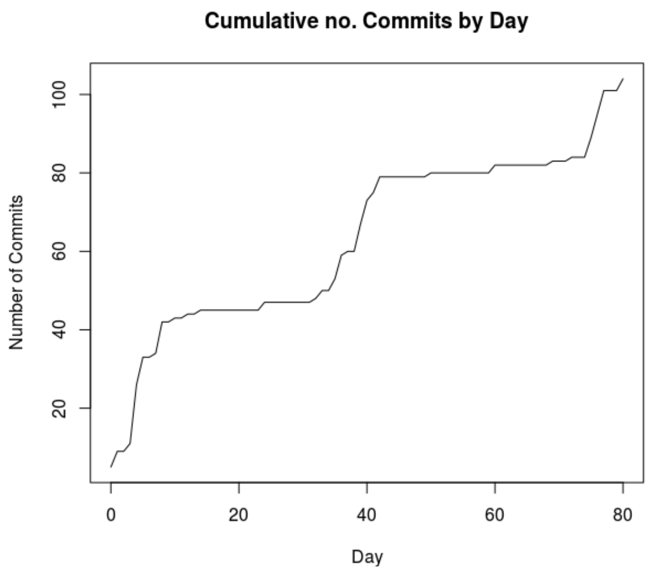
Team 1:



Team 2:



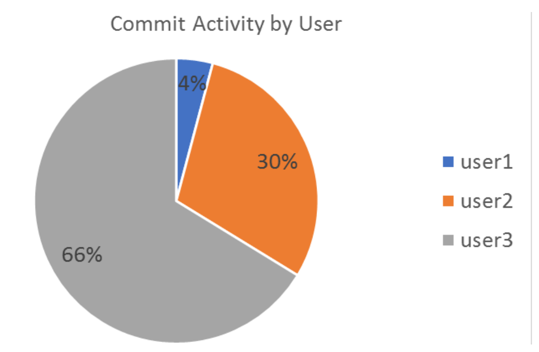
Team 3:



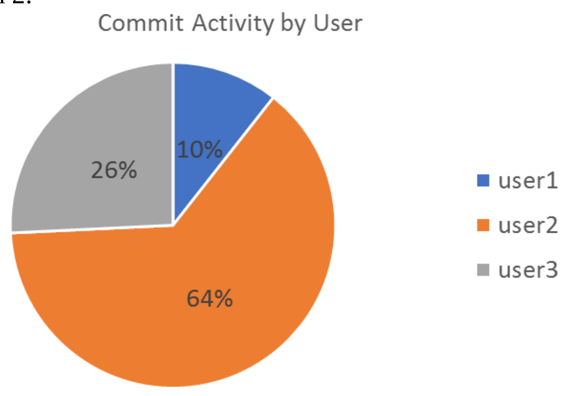
K. Commit Activity Per User

The figures below displays the numbers of commits per user. Ideally, each user should have the percentage of commits. For each group, we determine the difference between the maximum percentage of commit activity among the group's users and the minimum percentage. If this difference is greater than the expected proportion of activity for a given user (e.g., greater than 25% for a group of 4 users), it's considered a sign of an absent group member. Group 3 performed best, with the maximum percentage of commit activity being 40% from user 4, and the minimum percentage of commit activity being 17% from user 2. The difference between the two is 23%. The other two groups had larger differences between maximum and minimum number of commit activity percentage. The difference for Group 1 was 62%, and the difference for Group 2 was 54%. Since the difference between Group 1 and 2 is greater than 33%, we concluded this as a sign of an absent group member.

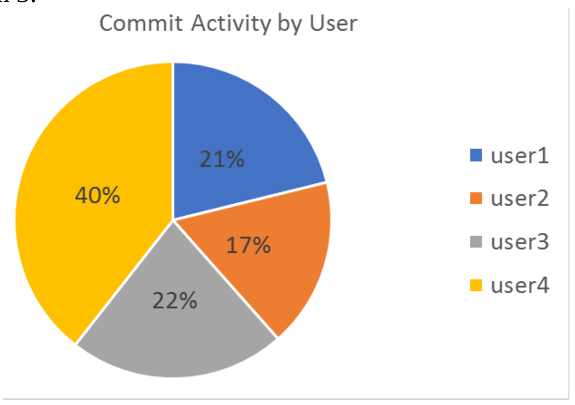
Team 1:



Team 2:



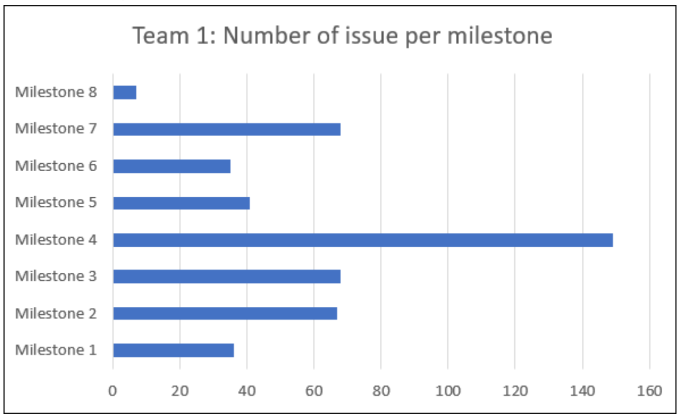
Team 3:



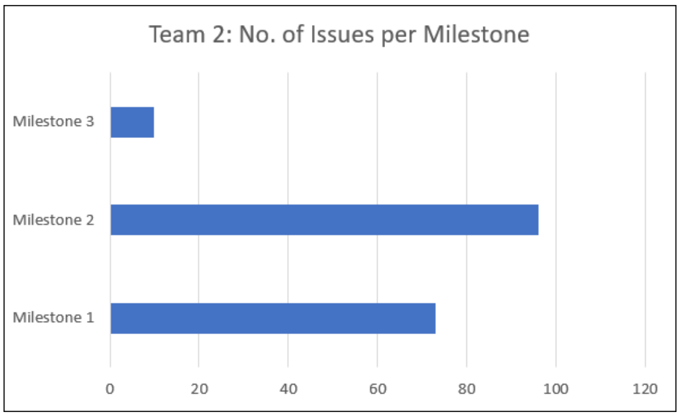
L. *Number of issues per milestone*

This feature is related to extracting number of issues assigned to each milestone. This will give us a clear idea of how much effort has been put on various milestones in the project.
From the graph, we can see that team 3 have only 3 milestones but two main, Milestone 1 and milestone 3, have almost equal number of issues assigned.

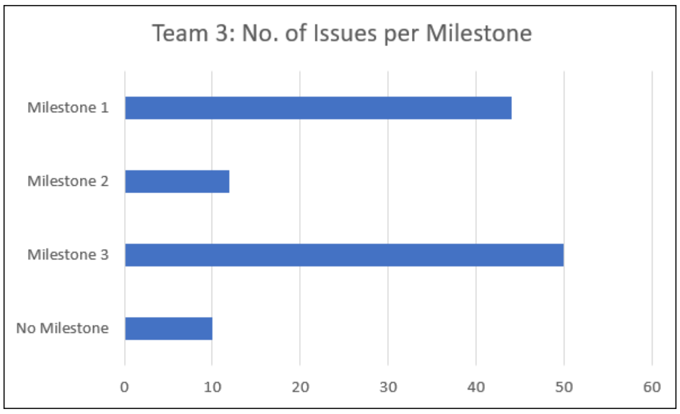
For Team1:



For Team 2:

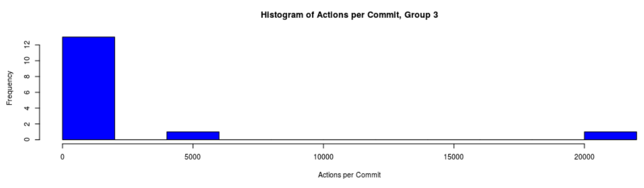


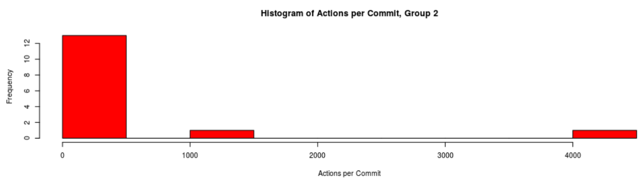
For Team 3:

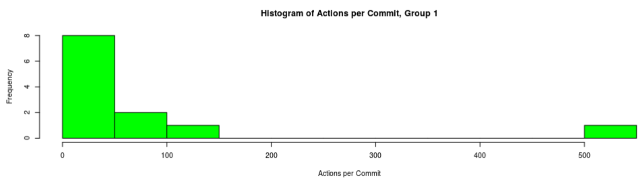


M. *Average Number of Actions per Commit by Week*

We define an action as a addition and/or a deletion made from a commit to the repository. The Figures 1,2, and 3 display the average actions per commit over a week long period. The red line represents additions made to the repository and the blue line represents deletions made to the repository. The two combined should equal the black line that represents the average additions and deletions per commit for that week. Each graph is over a 15 week period except group 1 because their repository wasn't created until later. Ideally a group should be pushing consistent small additions to their repository with little to no peaks. Groups should also avoid large deletions as this shows code they pushed was not effective. Also sections where the number of deletions exceeds the number of additions suggest the project isn't progressing well. All three groups exhibit a large spike in additions per commit at one point or another. This large spike occurs near the beginning of the project for groups 1 and 2, whereas for group 3 the spike happens around the March deadline. It is important to note that actions per commit are usually very low (between 5 to 70) for all of the weeks where the large spikes occur. Group 3 had the largest spike in additions per commit at around 20,000. Group 3 also had the largest deletions per commit at around 5,000 deletions per commit. Group 2 had the second highest spike of additions per commit at approximately 4,500. Lastly, group 1 had the lowest which was approximately 300 additions per commit at the very beginning of their project.
All three graphs show a heavily right skewed distribution as can be seen from figure #. The skewness values for the three groups are: 2.82 (Group 3), 2.89 (Group 2), and 2.40 (Group 1) which were calculated from the formula $\gamma`1 = \mu3/\mu23/2$ where $\mu3$ and $\mu2$ are the central moments. This suggests that while a large portion of the time the groups were doing a appropriately small number of actions for each commit, they did also sometimes perform a notably large number of actions per commit. For Groups 3 and 2 this is a bad smell in terms of uneven commits. However for group 1 we know that the largest number of actions per commit was performed at the creation of the repository, so we have not necessarily found a bad smell for this group.







IV. Early Smoke Detection

N. Milestone Completion trend

Analyzing the milestone completion date and due date of milestone. We found out that many groups missed the due date and the milestone was closed much after the due date. There are may be several reason for this like the members did not work seriously for the milestone completion or did not update the status of milestone completion in the repository.

27) Feature Detection

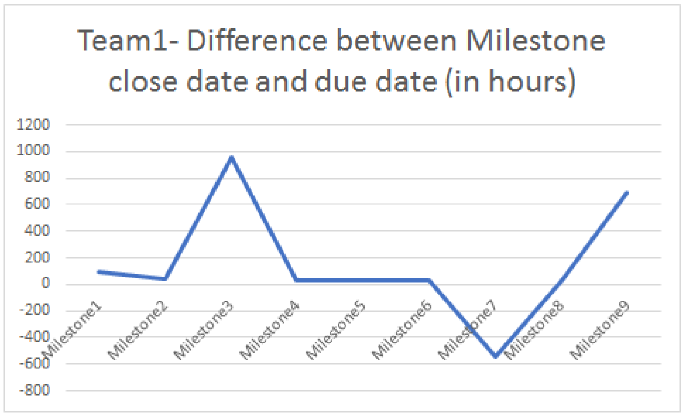
The data for each milestone was extracted for the groups and contained details like milestone_id, milestone_name, created_date, due_date, closed_date. There was some milestone without due date for which we manually put due date as per the month name in milestone.

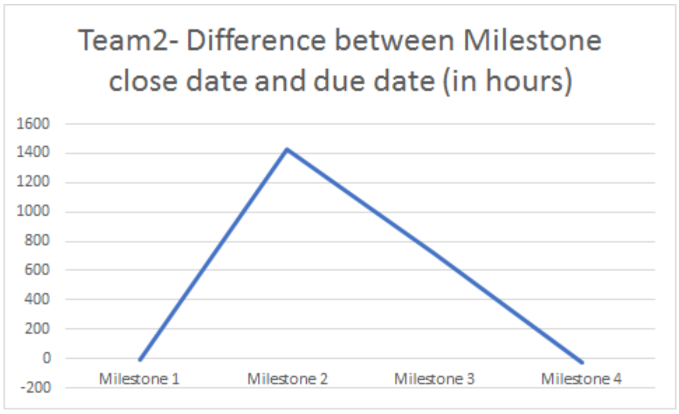
28) Early Smoke Detector

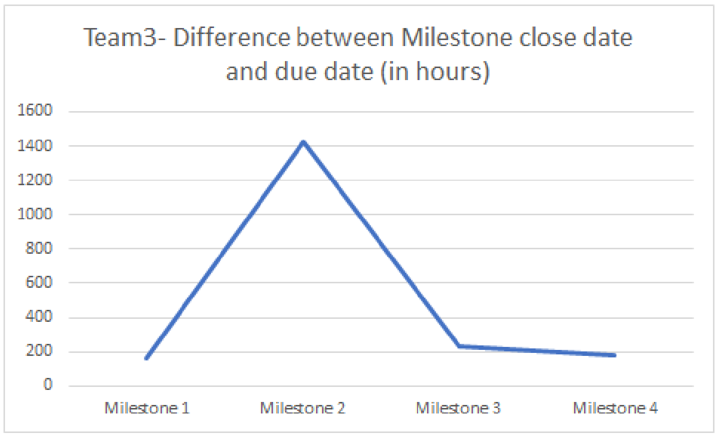
To be able to tell if the project will further miss the deadline for project milestone, first 50% milestones provide good inference. If the team has missed first 50% of the milestones, then we can say that it is going to experience delay in future deadlines too.

29) Early Smells Result

Following graphs show the trend of milestone due date and milestone close date in team's repositories







O. Overall commit trend

Analyzing the overall commit trend of the groups we found that members who show little or no contribution to the repository will continue to do so in the future.

30) Feature Detection

We got the data by visiting the repositories of the teams we were analyzing. We noticed that GitHub shows the contribution of each member in the overall repository by graphs of number of commits and also show the lines of code committed by each member.

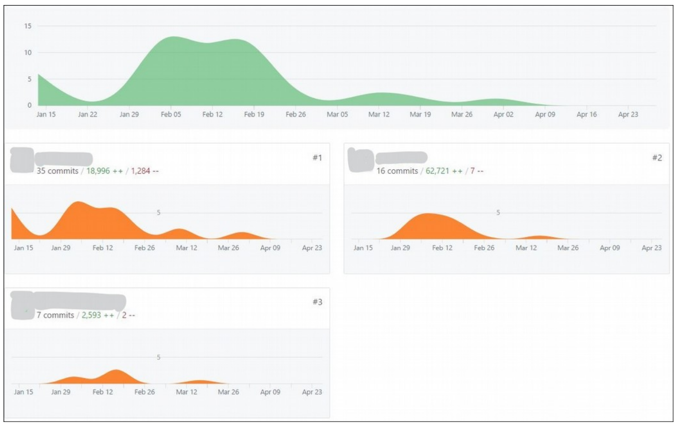
31) Early Smoke Detector

Commits by individual members provide a good factor for smoke detection. We found that members who do not commit much during the first month of the project, are more likely to do so in future. Analyzing the activity of members can provide good inference of the future. In such cases the other team members can talk to each other and take the help of instructor to ensure equal contribution from each member to have a smooth journey throughout the project.

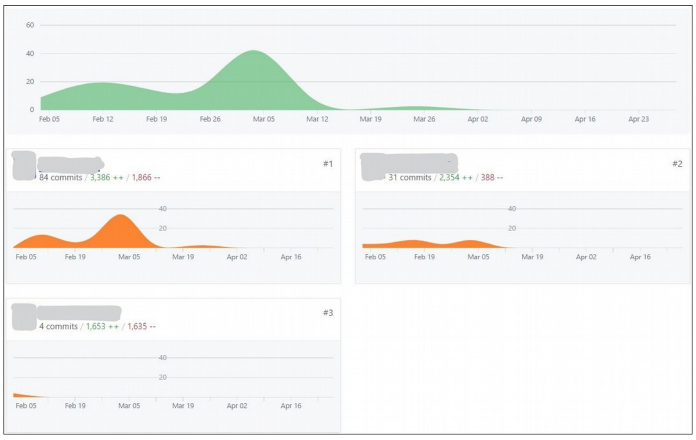
32) Early Smells Result

Following graphs taken from GitHub show the trend of overall commits for each member.

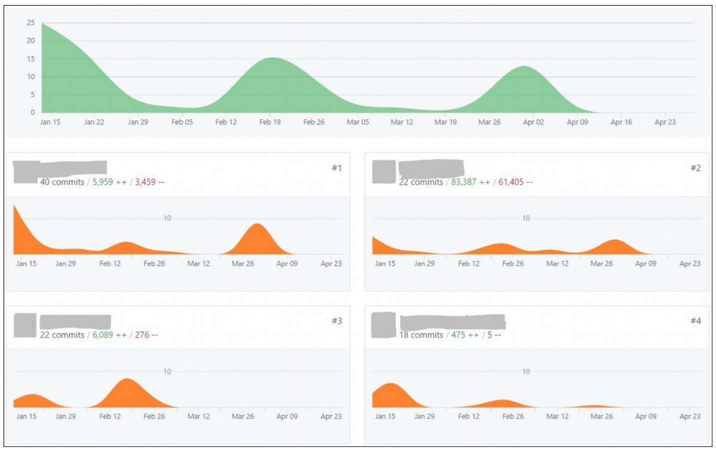
Team1:



Team2:



Team 3:



V. Conclusion

From the above analysis, it is clear that there are various factors which can be help find bad smells for our project. Almost all the repositories we analyzed had some or the other type of bad smell. Our study in software engineering has taught us how to deal with these bad smells practically and we are very sure that this will be very helpful for all our future projects.

Acknowledgment

We would like to thank Dr. Menzies for providing us the code for gitable.py and being a wonderful professor.

References

[1] Report Refernce- “https://github.com/stutinanda/CSC-510-Project2-badsmells/blob/master/May1_report_GroupD.pdf”

[2] Report Reference-“https://github.com/cleebp/csc-510-group-g/blob/master/may1/report/report.pdf”

[3] GitHub Developer. API.- "https://developer.github.com/v3/".s

[4] Report reference- "https: //github.com/CSC510-2015-Axitron/project2", 2015.