Project 2: A report on how to detect bad smells

Sukriti Sharma
North Carolina State University
Raleigh,NC
ssharm18@nscu.edu

Arjun Thimmareddy
North Carolina State University
Raleigh,NC
athimma@ncsu.edu

Ankit Murarka
North Carolina State University
Raleigh,NC
amurark@nscu.edu

Sushma Ravichandran
North Carolina State University
Raleigh,NC
sravich@nscu.edu

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; H.4 [**Information Systems Applications**]: Human factor; D.2.8 [**Software Engineering**]: Metrics—*observations, convenient measures*

General Terms

Observation, Case Study, Metrics

Keywords

Github, Milestones, Issues, Events

1. DATA COLLECTION

We collected a set of features from GitHub for each team. The data we collected is:

- Issues
  - Issue ID
  - Event performed
  - Label Name (If the event is a labelling event)
  - Assignee (If the event is assigning)
  - Assigner (If the event is assigning)
  - Time of occurrence of the event
  - The user who performed the event
  - The milestone, the issue is related to
  - Closing time of the issue
  - State of the issue (open or close)
- Commits
  - Commit id
  - The author who performs the commit
  - Time of commit
  - Message of commit
- Comments
  - User who comments
  - Comment ID
  - Issue ID the comment is for
  - Body of the comment
  - Creating time of the comment
- Milestones
  - Milestone ID
  - Milestone title
  - Milestone description
  - Milestone creation date
  - Milestone due date
  - Milestone closing date
  - Creator of the milestone
  - Number of open issues belonging to the milestone
  - Number of issues belonging to the milestone that are closed
  - State (milestone is open or closed)

1.1 Methods

We first selected three groups, other than our own which we would be using for our analysis. We selected groups who had raised a considerable number of issues. Next, we used gitable.py, provided by our professor to fetch the data. We added code to gitable.py to fetch additional data from the issues API, and to fetch data from GitHub APIs of other pages like commits, comments .
We then preprocessed the data to anonymize the group and user names. For the purpose of anonymization, we mapped each group with an ID defined by us, and each user with an ID defined by us. While collecting the information from the commits API, we realized that few users had not configured their GitHub login Ids , due to which the login Ids were missing. We handled this scenario by mapping their user names in this case with their Id. After performing anonymization, we inserted the information in MongoDB databases.

2. FEATURES

2.1 Patterns in Comments added by each Team Member

We felt that discussion and interaction is an integral part of the development process. And even though there are daily updates and weekly meets, its very important to keep track of others work and at the same time update others about one's work on a regular basis. This could be best achieved by commenting on issues and discussing about the progress and roadblocks.

2.2 Pattern of Commits over time

This feature collects data on the pattern of commits done by the users. This pattern allows us to draw conclusions on the active period of development and transitive period of development where the developers are not actively involved with the project. This can be considered as the downtime of the project. This allows the project manager or the scrum master to identify the downtime of the project and seek updates when required to assure timely delivery.

2.3 Distribution of Commits amongst Team members

This feature detects, if the work distribution between the team members is even. We determine this by tracking the number of commits by each member of the team. The results are used to analyze if the work distribution is even and if a particular member is not burdened with most of the workload while others are not actively involved.

2.4 Patterns in assignees commenting on Issues assigned to them.

The idea is to observe and identify patterns about assignees getting involved in the discussion happening on the issues assigned to them. This helps in identifying if the assignees are participating in discussions based on the issues assigned to them.

2.5 Modification to Assignees of an Issue

This unusual collection of statistics paints a key picture on the management. This gives us the statistics on the number of modification to the assignees of the issue has been through. This gives us an idea on the whether the management recognizes and prioritizes work. Sometimes too much shuffling of resources to accommodate new tasks can delay issues resolving time.

2.6 Patterns in Issues assigned to each Team Member

This feature is used to track the number of Issues assigned to each Team Member. Knowing the spread of issues amongst users will help in identifying the distribution of work amongst the users. If a certain user has no, or close to no issues assigned to him then there is a good chance that his contribution towards the project is either minimalistic or hasn't been carefully tested by others. Hence this feature helps us identify two possible deviations from Agile.

2.7 Number of issues under each milestone

The number of issues under each milestone got form git-hub allows to identify the key milestone in which much of the features or the effort was put in by the development team. This feature allows the manager to ensure the target is not something which cannot be achieved on time. It allows him to identify "The team doesn't bite more than it can chew". This ensure to follow the principle of under promise and over deliver which builds a good relationship with the customer and develop a good long term relationship with the customer.

2.8 Milestone duration and completion

The milestone duration is a key feature where we collected data over the course of the project about the duration of each milestone and the milestones which were not completed on time. This feature gives an idea of the how the milestones created over the course of a project is assigned the deadline or the due date. Ideally milestones created during scrum should be small and should span over a small period of time as rapid delivery to target and meet the customers demand. An ideal scrum master would ensure the milestones are small and appropriate in number to allow easy tracking. An exhaustive list off milestones are also not good as it dilutes the purpose of a milestone.

2.9 Duration - The issues were open

This feature collects statistics on the number of days, the issues were open. This recognizes the issue which were unattended for a long time or the issues which were tough and took a long time to solve. This allows to identify the duration - the issues was opened in general during each phase of the project.

2.10 Pattern of issue creation

This feature collects data on the issues created over time. i.e the number of issues created over the course of time. Using git we were able to collect this data over the project duration. The number of issues created per week gives us a view on the effort and active participation of the team on the project. This gives an idea of ramping up the project delivery or poor coding practise.

2.11 Categorization of issues under milestones

The categorization of issues under the milestone is another statistics collected to identify issues which are not part of the milestone and issues which are part of the milestone. Ideally issues are assigned and created after the scrum meeting, once done these issues are due to be delivered for the next sprint or the next weeks release cycle. Its very ideal to categorize each issue into one of the milestone. This allows the scrum master to ensure the deliverable is marked under the specified milestone and is easy to track in the ocean of issues where the priority is not ratified initially.

2.12 Pattern of issue closure

This feature collects data on the issues closed over time. i.e the number of issues being closed over the course of time. Using git we were able to collect this data over the project duration. The number of issues closed per week gives us a view on the effort and active

participation of the team on the project. This gives an idea on the ramping up of the project delivery or the poor issue closure situation where there are no active delivery being made on the project.

3. RESULTS ANALYSIS

3.1 Comment distribution per user

The comment distribution graph in Fig 1. is an indicator of the percentage of comments written by each user in the group. This helps us identify the amount each user has communicated during the course of the project. Ideally, all users should be actively involved in communication and the percentage of comments should be equally distributed amongst the members. We find the graph to be unusual if few users have not commented significantly, as that indicates that they did not participate actively

3.2 Commit distribution over time

The commit distribution graph in Fig 2. is an indicator of the amount of commits done by the group over the project course. It allows us to detect the abnormality in terms of unusual spikes and drops in the graph.Ideally the graph should have a uniform or a linear growth. Unusual spikes indicate the project was not constantly in progress. Also its not a good practise as there are chances of loosing developed source code. The developer has to make sure, the repository is updated with the latest code so as to allow more contribution and less synchronization issues with other users. This is also one of the reasons for the merge conflict issues, where the user commits stale source code into the repository.

3.3 Commit distribution per user

This graph in Fig 3 is an indicator of effort and willingness of each user to contribute to the project.The graph ideally should have equal distribution in the pie chart where each user contributes 25 or more percent give or take 5 percent. The graph is clear indicator of the effort put in by each user or the activeness of productiveness of each user. Less than usual commits by each user is an indicator of communication failure and tending towards synchronization issues with team mates. This can have adverse affect during the production migration phase of the project and UAT phase - where the clients can get stale product with not all features in it due to low frequency off commits.

3.4 Distribution of Issues with and without assignee comments

The fig 4. is an indicator of the distribution of milestones on which the assigned user did not even comment. This graph though naive is great indicator of the poor communication where the user assigned the issue, never made an attempt to comment on the issues. Ideally the user whom the issue is assigned should monitor the issue and keep the issue updated with his comments. Any failure to do so is an indicator of the poor communication.

3.5 Assignee Modification Per Issue

The number of Assignment Modifications performed on an issue reflects how the person responsible for a particular portion of wok was continuously changed. This shuffling of task force indicates poor planning. In Figure 5 shown, Greater than 8 modifications goes to show the amount of mis-management that was caused during the issue resolution process.

3.6 Issues assigned per person

The issue distribution graph per user in Fig 6. is an indicator of the number of issues assigned per user. Ideally, each user should work almost equally on issues. We find the graph to unusual if maximum issues were assigned to one or two people in the group, and the distribution is not uniform.

3.7 Issue Distribution per Milestone

The number of Issues associated with a milestone is an important metric during statistical analysis. It goes to show the number of problems resolved over a milestone which backs the idea of Agile Development. The graphs under Figure 7 show that not all milestones have had issues under them. This indicates that there were milestones created without a purpose.

3.8 Milestone Duration

It is important for a milestone to be closed before its due date. The graphs of Figure 8 for each user group shows the length of each milestone they created. The red line indicates the due date of the milestone. If a milestone extends beyond the due date, it reflects poor practices by the respective members. It also helps us to notice any bad smell early in the development process itself because as soon as the project does not meet deadlines it is indicative of under performance by the respective group.

3.9 Issue Duration Distribution

The graph shows the four groups and the duration of their issues (in days). We see that most of the issues do not cross the 10 day mark. Agile supports deliverables at short intervals of time. Hence, an issue that is closed after a long duration of time indicate poor Agile practises and bad smell. We have gathered statistics of the same and represented them in Figure 9 where we have categorised the issues based on how long they have remained open.

3.10 Issues created over time

The issue distribution graph over time in Fig 11. is an indicator of the number of issues created weekly over the duration of the project. Ideally, we expect that as the users are committing code, they will keep identifying issues. Since the groups are following agile, they should ideally create issues after a sprint, and on completion of those, create more issues. Hence, we would expect to see issues created throughout the duration of the project at regular intervals. We would find the graphs to be unusual if issues are not created uniformly , and if all issues are created only at one point of time.

3.11 Issues under Milestone Distribution

Figure 1: Comment distribution per person

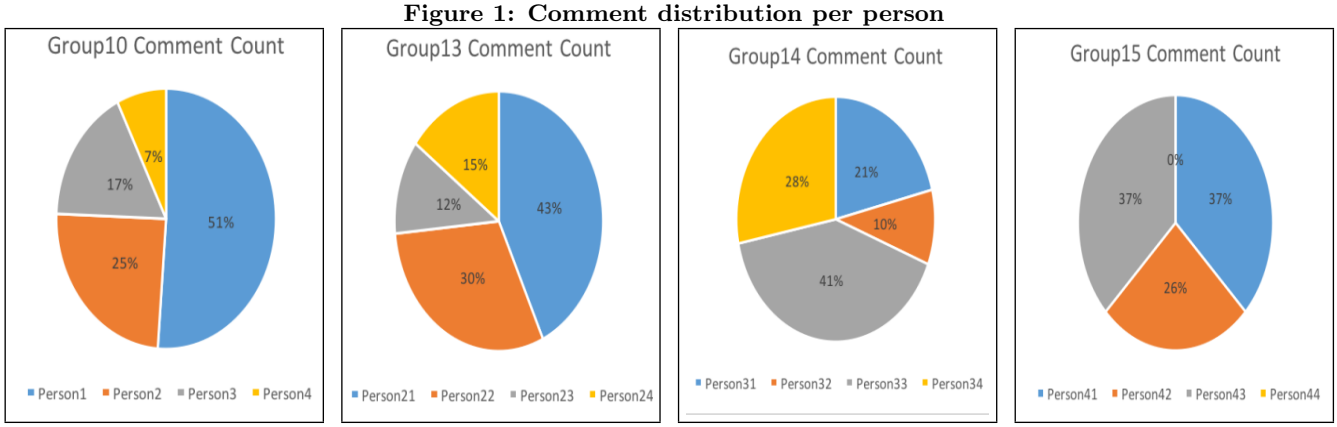


Figure 2: Commit distribution over time

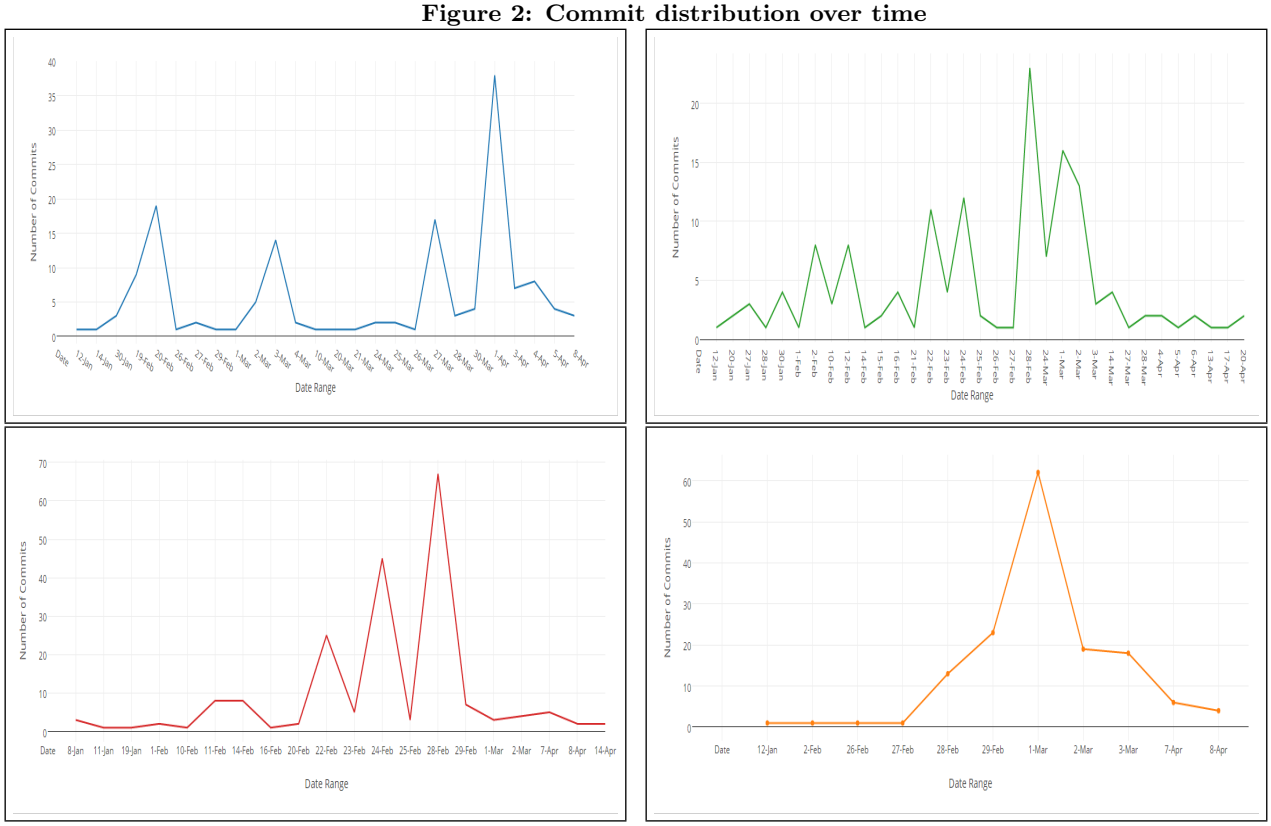


Figure 3: Commit distribution per user

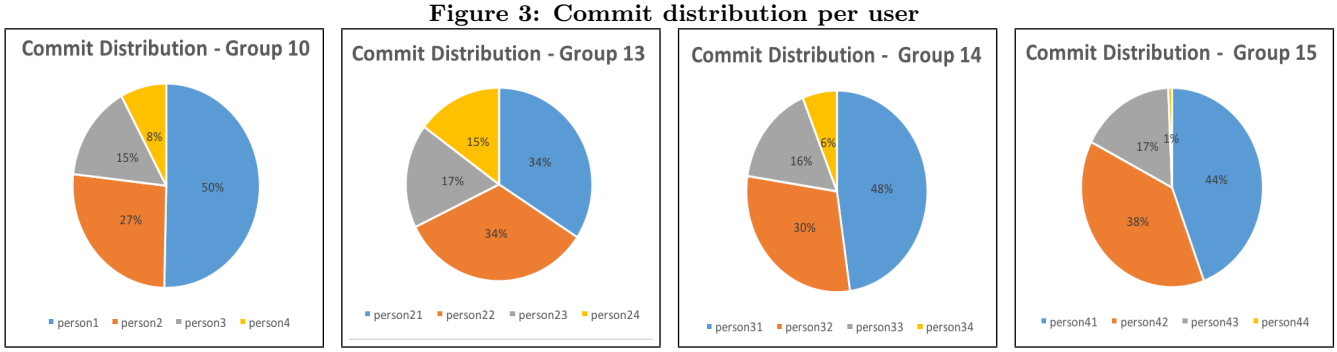


Figure 4: Distribution of Issues with and without assignee comments

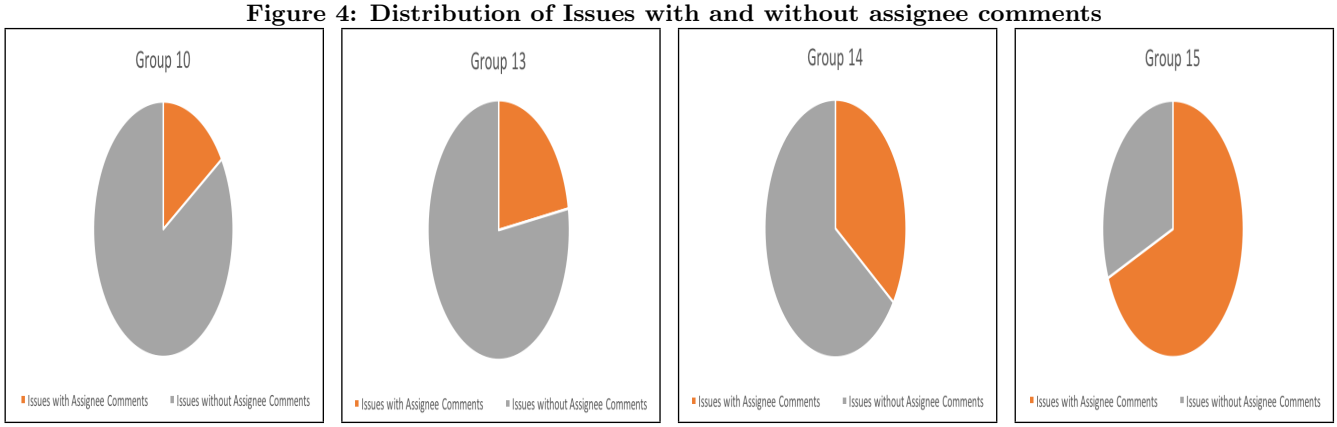


Figure 5: Assignee modifications per Issue

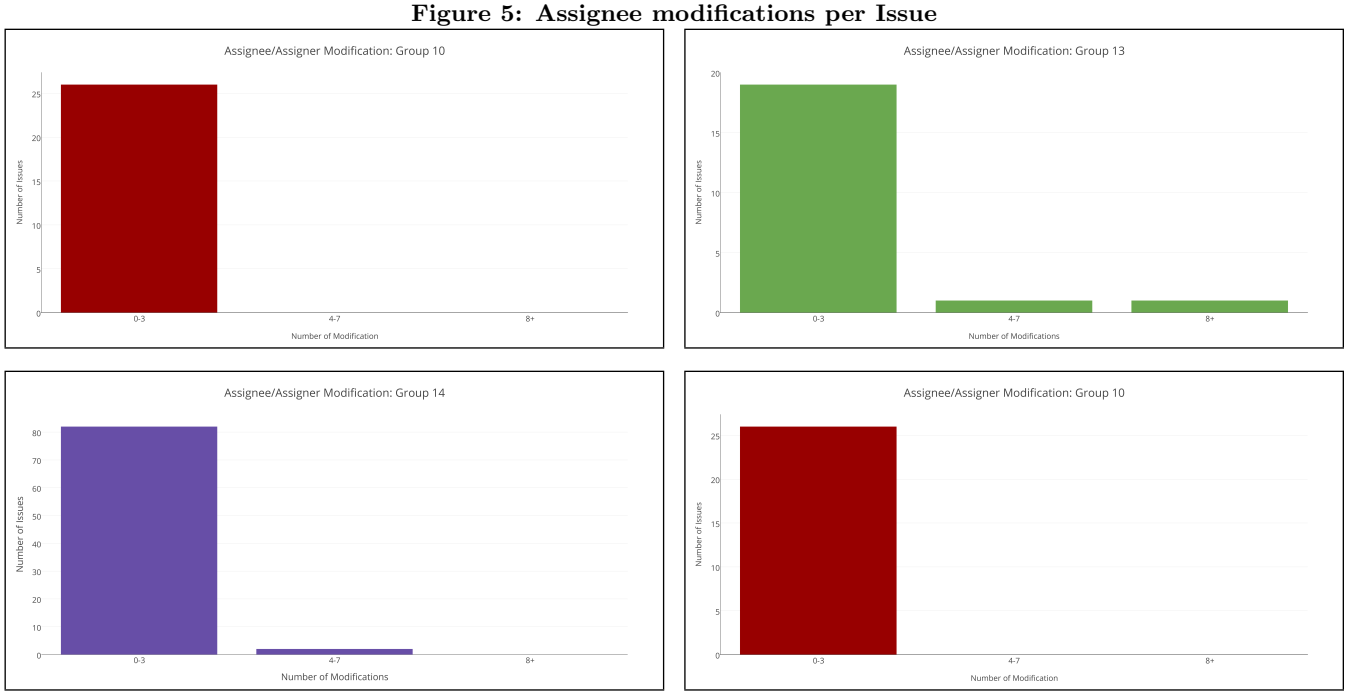


Figure 6: Issue distribution per user

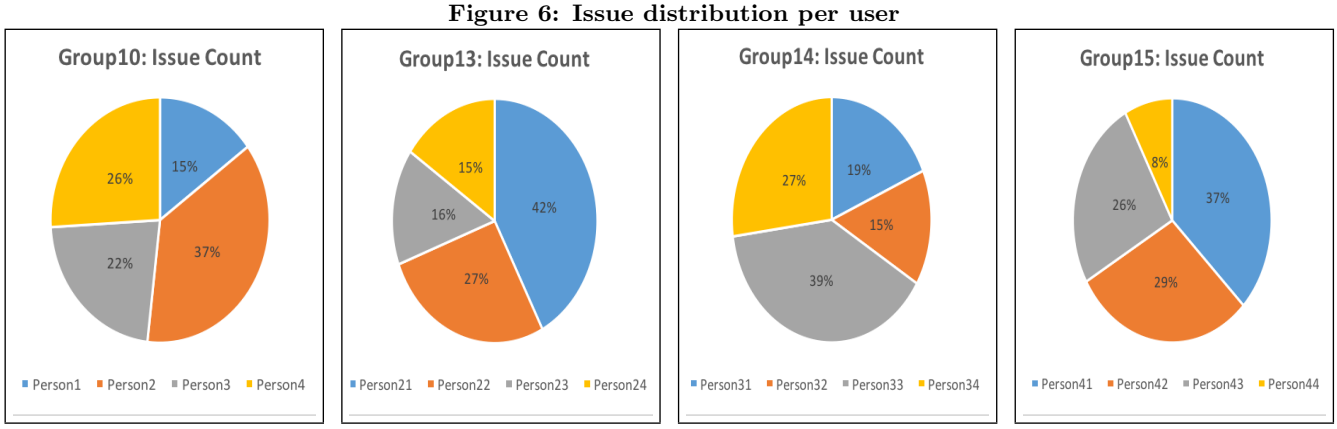


Figure 7: Issue distribution per Milestone

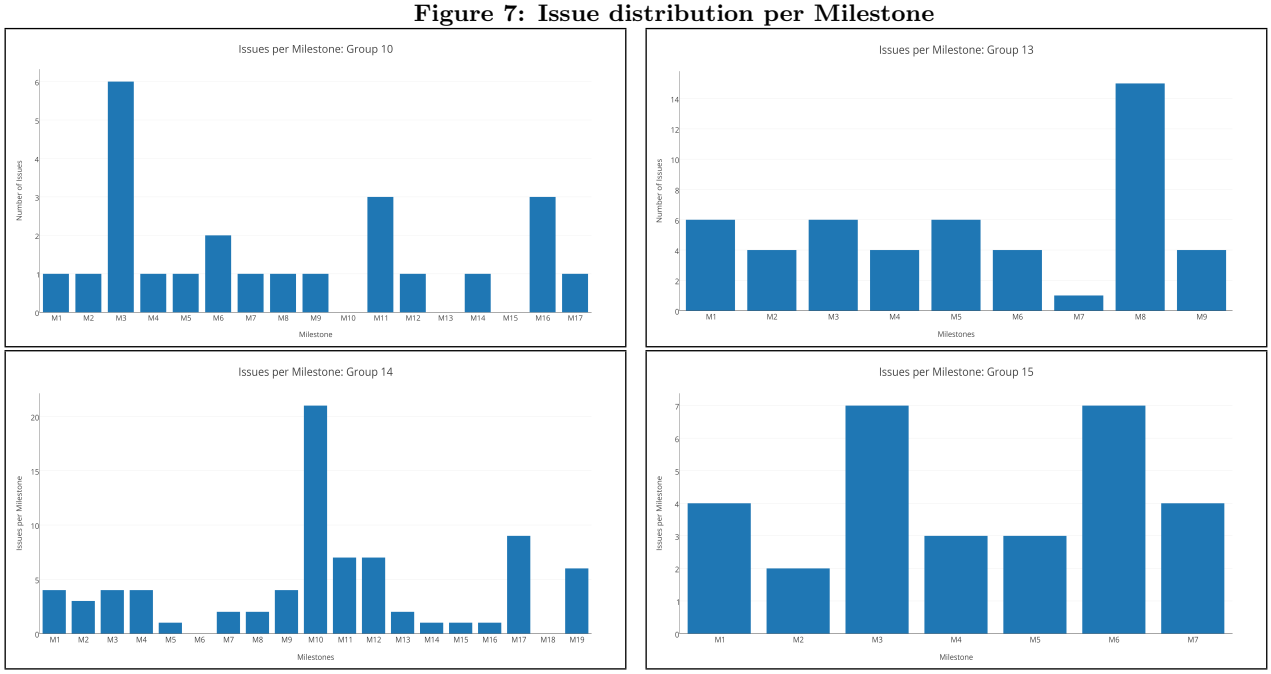
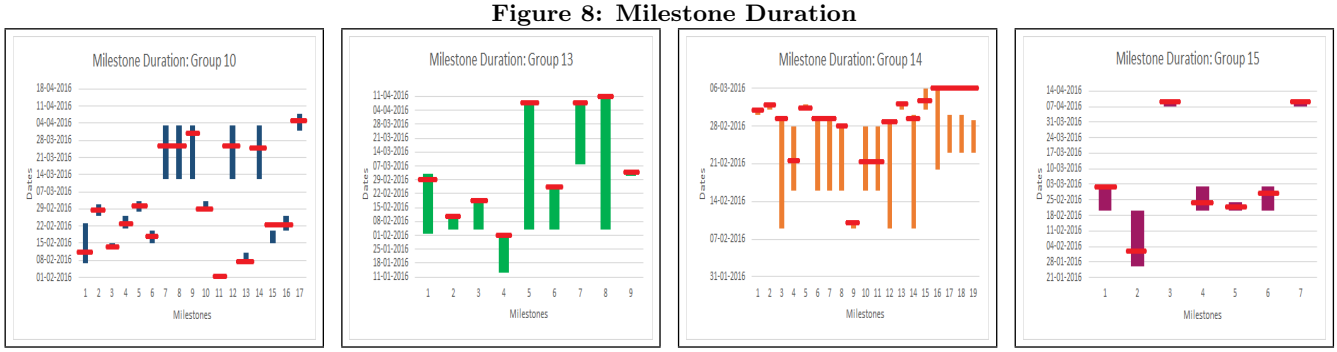
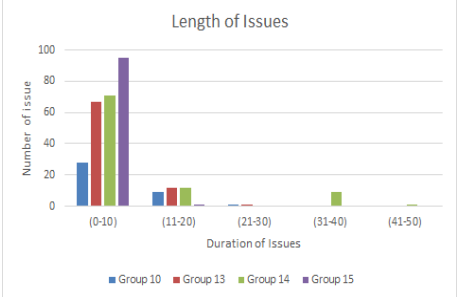


Figure 8: Milestone Duration



Issues are a good way of keeping track of tasks, bugs and recent modifications to the projects. It is important to note that git-hub allows issues to be categorised under milestones. The graph as shown in Figure 12 associated with this shows the number of issues under milestones (in purple) and those that do not belong to any milestone (yellow).

Figure 9: Issue Duration Distribution



3.12 Issues closed per week

The fig.10 is an indicator of the throughput of the team during the project course. It clearly indicates the amount of effort put in by the team during the deliverable period. The more issues being closed per week, the team is on track for the delivery of the project. The graph should be uniform and linear in the way the issues were closed over the period. Unusual spikes is an indicator of the bulk update which is wrong in terms of a agile methodology of the project.

4. BAD SMELL DETECTORS

After carefully extracting the information from the database, we identified a few potential bad smells. We compared 4 groups and analyzed if they successfully avoided the bad smells or not. The motive was to identify bad smells based on common Agile practices in Software Development.

4.1 Poor Communication

One of the most basic forms of Bad Smell detector is when the members of a group are not communicating

Figure 10: Issues Closed Per Week

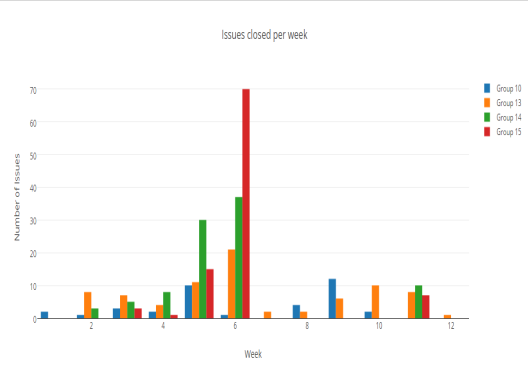
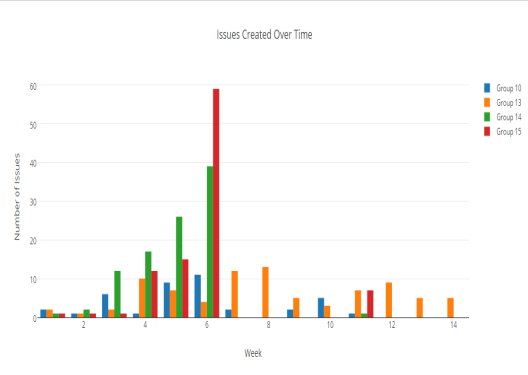


Figure 11: Issues created overtime

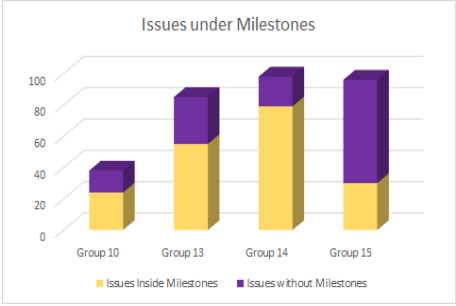


strongly during the course of a project. It is important that each member of the group is heard and understood before and during the development of a project. Communication avoids a lot of faulty and irrelevant assumption at a later stage. We have chosen to identify Poor Communication from a group's github repository using the following metrics:

- Assignee modification of Issues
- Comments per group
- Issues with assignee comments
- Issues with/without comments

We have elaborated the importance of each of the statistics below. We found the number of times an Assignee was modified for an Issue. Although a number of groups did not assign/reassign more than 3 times, we did find some extreme cases. We chose to set the limit at 3. If the count of modifying an Issue Assignee was greater we declared the scenario as bad smell owing to Poor Communication between the users in the group. Comments are the direct indicator of any form of communication through github between users while developing a product. We used pie charts to represent the percentage of communication with regard to comments per user in each group. Assuming a 4 member team, we regarded low percentage of Issues in github assigned to a particular user when a bug is found or when a modification/fix of code is required from that user's end. This person is the assignee of the Issue.

Figure 12: Issues under Milestone Distribution



Whenever an Issue has an Assignee, it is expected by the Assignee to inform the community if he has seen the message or if the Issue is resolved and how. Failing to do so, reflects bad communication between the members. We gathered statistical information to prove the same. We conclude that an assignee should have commented at least a few times under the Issue he has been assigned. Issues are a way to inform the other users of a group that a bug has been found or if a task has been accomplished. It is very similar to emails but can be shared by all the members of a group. It is one of the most direct indicators to communication among the group members on github. Issues without any Comments on them demonstrate how inadequate the communication between users have been and proportionately exhibits bad smells.

4.2 Poor Work Distribution

This bad smell describes uneven distribution of work among the team members. This can result in some members being overloaded with work while others taking it easy without any substantial work. To avoid this, planning and discussion should be done before assigning or taking up tasks. This can prove to be detrimental to the general morale of the team and to the members who are facing the maximum brunt. Some of the features which relate to this bad smell and help identify it are:

- Commit distribution per user
- Comment distribution per person
- Issue distribution per person

We analyzed the bad smell using these features by capturing the percentage distribution of commits, comments and Issues handled by team members. An unusually high percentage for a particular member as compared to other team members was considered a bad small.
Through an uneven distribution of commits we concluded that the code wasn't written evenly. While some members contributed more towards development others slackened in the most crucial part of the process. Through an uneven distribution of comments we concluded that the discussions were mostly one sided. While some members showed interest in discussing the features and issues, others didn't show much enthusiasm in the discussions.
Through an uneven distribution of issues we concluded that while some members faced all the roadblocks and/

bugs, there wasn't enough work done by others or their work wasn't properly tested or monitored.

4.3 Poor Milestone Usage

Poor milestone usage refers to the low usage or non usage of the milestone feature. The purpose and usefulness of the milestone are to track every single deliverable of the project and the work associated with that particular deliverable. This gives a good estimate of the effort put in by the team for the deliverable. The stats collected on the milestones can be used to gather very important statistics. The below features collected by us allows us to track the bad smells.

- Issues per milestone
- Issues without milestones
- Issues per person
- Milestones incomplete

From the above graphs we can clearly identify the groups which had poor milestone usage. The number of issues per milestone is one of the factors, any milestone created is symbolic of a deliverable and its should have issues associated with it. If there are no issues assigned to the milestones, it looses its essence of representing a deliverable and the work associated with it. The work associated can be anything from an enhancement, bug or a new feature being implemented. The issues needs to be under the categorized milestones. If the issues are not under the milestones, these are stray issues without any predefined delivery date and can stray along for a long time without any one acting on it. Hence it is very necessary to include the issue under milestones.
Usage of milestones although important, the right usage of the milestones is also the key. The milestones created should be concise and should be coarse grained to a limited extent so that they represent a classifiable deliverable rather than the whole project being represented as a deliverable. If the milestone has too many issues associated with it, it defeats the purpose and scrum master cannot track the large milestone with too may issues. Smaller concise milestones can do wonders as the scrum master can achieve granularity of the issues being tracked. We in the bad smell results very clearly describe the results in terms of each group.

4.4 Intermittent Work Periods

This bad smell is all about work being done intermittently. This can be concluded if the graph shows unusual spikes and valleys. This means that there are certain stretches of days when there is a lot of activity by the team members and then there are other days in between these stretches which shows a lean period showing minimal or no activity by the team members. Some of the features which relate to this bad smell and help identify it are:

- Commit distribution over time
- Issues created over time
- Issues closed per week
- Milestone duration

While commits distribution clearly identifies lean patches when there was no work done and stretches where a lot of work was done, issue creation and close frequency also plays a crucial role in identifying irregular patterns in the work schedule followed by the group members in a team. Issue creation distribution state how often the code is monitored and bugs are found. Issues closing time determine how quickly these open issues are looked into and resolved. Milestone duration can also lead to a bad smell if the milestones are either too long(which is not a good agile practice) or the milestones are closed after the due date.

4.5 Poor Planning

Poor planning bad smell is derived from a mix of lot of graphs where the key is to identify the factors which led to delay in the project being delayed or unmanageable. Some of the factors like Inactivity, less milestones, assignees changing multiple times, milestones with long durations, issues closed after milestones as it could not be finished within time, abnormal issue distribution to developers.
The graph in fig.8 is an indicator of milestone duration where milestones spanned over many months and were not able to be accomplished before its due date. The fig 8 clearly indicates this to be a poor planning where the team cannot accomplish the task on time. So its very important to estimate the project duration accordingly.
The Fig 5 is an indicator of the number of assignee changes for the issue before its completion. It indicates the team could not narrow down on the cause and the issue kept on circling around the teammates, along with the issue completion duration gives a good picture on why the issue couldn't be solved on time for the deliverable to go with all the requisite bugs/enhancements.
The issue distribution graph in Fig 6 indicates the distribution of the issues over the team. The unusual planning and assignment of a high number of issues to a person would lead to low delivery and create situation of high dependency which might roll the project off track due to absence or no delivery of a single individual.

5. BAD SMELL SCORING

- Assignees Assigner modification : We assign bad smell score of +1 for any modifications on the assignee more than 3.
- Comment Contribution : We assign a bad smell count of +1 for any user whose comment contribution is less than 15 percent.
- Commit per day : We identify the inactivity by the team for close to a week and assign a bad smell score of +1. For Spikes in the number of commits more than 50, we assigned a bad smell score of +2.
- Commit Distribution per person: We identify the commit distribution or contribution by each person. If any team member has less than 15 percent contribution. We assign bad smell score of +1.

– Life span of issues : If issues spanned in the range '>' 20 and '<' 30 they get a score of +1. If issues spanned in the range '>' 30 and '<' 40 they get a score of +2. If issues spanned were '>' 40 they get a score of +3.

– Issues with/without Assignee comments Distribution : If the percentage of issues with assignee comments is < 25 percent of the pie chart then we assign a score of +3. Assign +2 for percent between 25 and 50 and +1 for percent between 50 and 75.

– Issue Count per person : We identify the issue count distribution and assign a bad smell score of +1 for any team where the individual team member contributed to less than 15 percent of the issues.

– Issues closed per week : We identify the issues closing pattern, if the issues closed are more than 50 per week. We assign bad score +3 accordingly. If its within the range of 20 to 30, we assign a score of +1 and 30 to 40 we assign a score of +2.

– Issues per milestone : If we don't see any issues under milestone. We assign a score of +1 as milestones should contain some issues, enhancements under it.

– Issues not under milestone : If more than 25 percent of the issues are not under any milestone. We assign a bad smell score of +1.

– Issues created overtime : If we see any unusual high issue creation of > 50 per week then +3, > 30 then +2 and +1 for > 20.

– Total comments distribution per user: If the contribution to the comments by each individual is less than 15 percentage then we add a bad smell score of +1.

– Milestones crossing Due date : For this scenario, we divide milestones crossing due dates by the total milestones and multiply by 10. This is the bad smell score assigned to the group.

6. BAD SMELL RESULTS

We calculated the bad smells per feature for each of the five groups using the scoring format we described earlier. This gave each group a Bad Smell Counter for each score per bad smell. All results of the groups per feature have been listed in the tables below. We wanted to analyse the performance of the groups on github and try to find out if an early detection of Bad Smells is possible. We also tabulated the overall Bad Smell Counter for each group to find out if there was any cumulative prediction of any group not following the Agile Methodology.

As shown, Group 13 performed best for the features we selected per group and the scores we added to them. The cumulative score chart clearly depicts the same. We also find that a number of features have contributed to the relatively higher scores of the other groups. Some of those features which can help us identify Bad Smells early in the development cycle have been described in the following section.

Figure 13: Bad Smells per Group

Bad Smell 1: Poor Communication				
	Group 10	Group 13	Group 14	Group 15
Assignee Modification of Issues	0	2	1	1
Comments per Group	1	1	1	1
Issues with Assignee Comment	3	3	2	1
Issues without Comments	3	3	2	1
Total	7	9	6	4
Bad Smell 2: Poor Distribution				
	Group 10	Group 13	Group 14	Group 15
Commits Distribution Per User	1	0	1	1
Comments Per User	1	1	1	1
Issues per Person	0	0	0	1
Total	2	1	2	3
Bad Smell 3: Poor Milestone Usage				
	Group 10	Group 13	Group 14	Group 15
Issues per Milestones	3	0	2	0
Issues under Milestones	0	0	0	1
Milestone Duration	6	1	3	6
Total	9	1	5	7
Bad Smell 4: Intermitten Work Periods				
	Group 10	Group 13	Group 14	Group 15
Commit Distribution over time	3	0	5	6
Issues Created over time	0	0	3	3
Issues closed per week	0	1	4	3
Milestones Duration	6	1	3	6
Total	9	2	15	18
Bad Smell 5: Poor Planning				
	Group 10	Group 13	Group 14	Group 15
Issues per Milestones	3	0	2	0
Issues under Milestones	0	0	0	1
Milestone Duration	6	1	3	6
Total	9	1	5	7

7. EARLY WARNING

– Using the commits per week, we can diagnose inactivity for a long periods of time initially in the project indicates poor communication in the team as they do not interact and deliver on time.

– Using the stats on the issues closed per week, we can identify and track the number of pending issue being accumulated. If the number of issues exceed a threshold then we cannot achieve the delivery date of the project and can be caught very well before the deadline.

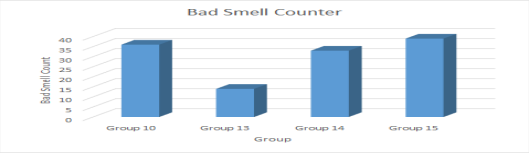
– Using the graph on the issues created over time, we can identify the spikes which indicates panic mode and creation of issues in one moment.

– Milestones surpassing the due date is a clear early indicator on whether the team is actually delivering on time initially. So that appropriate actions can be taken to prevent this from happening again.

8. EARLY WARNING RESULTS

We examined the 13 feature detectors that we defined earlier and picked Commits per Week(Fig 2), Issues

Figure 14: Overall Bad Smell Counter



closed per week(Fig 10), issues created overtime(Fig 11) and Milestone Duration(Fig 8) as our early warning detectors. The reason for choosing these as the early warning detectors are as follows:

- The commits over time distribution per week gave us bad smells in the very first couple of weeks when some of the weeks failed to commit more than the minimum number of times in a week (1).
- The issues created overtime should not exceed a threshold limit as that signifies poor planning. We again found a few bad smells early into the project when the teams crossed this threshold limit and created many issues in a single week.
- The issues closed in a week also should not exceed a threshold limit as that signifies delayed action. We found a few bad smells when the teams crossed this threshold limit and closed many issues in a single week.
- The milestone duration graph clearly indicated that a few teams missed their very first milestone. The milestones were closed after the due date. This smells bad because it indicates improper planning and estimation.

9. CONCLUSION

Github again has proved to be a efficient tool which not only simplifies the whole process of source code management but also using the data mining techniques we can easily mine the repository data to identify the cause and effects of a bad coding and management practises. The project management team can easily use these and identify the pros and cons of such practises. This can be used to avoid any such failures in future by identifying the traces much earlier in the process.