Project 2 : GitHub Bad Smell Detector

Yihuan Dong
North Carolina State University
ydong2@ncsu.edu

Niki Gitinabard
North Carolina State University
ngitina@ncsu.edu

Linting Xue
North Carolina State University
lxue3@ncsu.edu

Rui Zhi
North Carolina State University
rzhi@ncsu.edu

ABSTRACT

For project 2, we collected and analyzed GitHub data from 14 "Software Engineering Groups". We defined "Bad Smells" in these projects' behaviors including issues, milestones, commits and comments problems. We also built "Bad Smell" detectors based on our definition. In this report, we will talk about the data we collected, data anonymization, feature extraction, bad smell definition and detection, and case study. Further more, we will present a bad smell early detector model at the end of the report.

1. DATA COLLECTION AND FORMATTING

1.1 Overview

To look for signs of bad smell in project 1, we need to get GitHub usage data from all the groups. We took advantage of GitHub API and gitable.py code provided in the project description, modified the code according to our needs and pulled useful interaction data from GitHub.

For data collection, we collected data on multiple aspects, from issues, issue comments, milestones, and labels to all the commit information from each user. We will talk about what specific data we collect for each aspect in detail in the following sections. In order to organize and analyze data files from different groups efficiently, we first created a folder for each aspect. Then we store data from each group into separate CSV files and put the files into corresponding aspect folder. The number of data we collected for each aspect is shown in the following table 1.

1.2 Issues

The format of data we crawled for issues are shown in Figure 1.

The "assignee" is the identification number for the user assigned for this issue. "closed_at" and "created_at" are the timestamps when the issue is closed and created. "comments" is the number of comments under this issue. "id" is an identification number for this issue. "labels" stores all the labels that are attached to this issue. "milestone" stores the milestone that the issue is assigned to. "state" is the current status of the issue, either "open" or "closed".

Table 1: GitHub Data Stats: Total number of groups, issues, milestones, comments, commits and labels in project 1

Groups	Issues	Milestones	Comments	Commits	Labels
14	934	146	965	1752	119

```
{
"assignee":   {"id": 8457618},
"closed_at":   "2016-02-16T20:06:32Z",
"comments":    4,
"created_at": "2016-02-08T00:57:06Z",
"id":          132030613,
"labels":      [{"name": "data-collection"}],
"milestone":   {"due_on": "None"},
"state":       "closed"
}
```

Figure 1: Issues Raw Data

1.3 Milestones

Figure 2 shows the data format we collected and used from GitHub milestone page.

"id" is the identification of the milestone. "state" is the status of the milestone, whether it is "open" or "closed". "open_issues" and "close_issues" store the number of issues assigned to this milestone that are open and closed. "created_at", "updated_at" and "closed_at" record the time-stamp when the milestone is created, updated and closed. "due_on" is the due date of the milestone.

1.4 Comments

Figure 3 shows the data format we collected and used for comments. For comments, we only collected and analyzed one feature, which is the identification number of the user

```
[
  {
    "id": 1678631,
    "state": "closed",
    "open_issues": 0,
    "closed_issues": 1,
    "created_at": "2016-04-01T18:45:22Z",
    "updated_at": "2016-04-08T16:08:14Z",
    "closed_at": "2016-04-08T16:08:14Z",
    "due_on": "2016-02-01T05:00:00Z"
  }
]
```


Figure 2: Milestones Raw Data


```
[
  {
    "user": {
      "id": 1
    }
  }
]
```


Figure 3: Comments Raw Data

who wrote the comment. Hence we can count how many comments each user make.

1.5 Commits

Figure 4 is a list of features that we used in the analysis of commits: "commit.committer.date" is the time-stamp when


```
[
  {
    "commit": {
      "committer": {
        "date": "2011-04-14T16:00:49Z"
      },
    },
    "author": {
    "id": 1,
    },
  }
]
```


Figure 4: Commits Raw Data

the commit was made. "author.id" is the identification for the author in the repository.

1.6 Labels

Figure 5 shows the features we analyzed for labels.

We only used the name of label in multiple analysis related with label.

1.7 Contributors

Figure 6 shows the data format of GitHub contributor page: "author.id" is the identification for the user in the repository.


```
[
  {
    "name": "bug"
  }
]
```


Figure 5: Contributors Raw Data


```
[
  {
    "author": {
      "id": 1,
    },
    "total": 28,
    "weeks": [
      {
        "w": "1367712000",
        "c": 10
      },
      {
        "w": "1377712000",
        "c": 18
      }
    ]
  }
]
```


Figure 6: Contributors Raw Data

The "total" is the total number of commits made by current author in this repository. The "weeks" stores the number of commits the author made for each week. More specifically, "weeks.w" stores the time-stamp for the week and "week.c" stores the number of commits the author made in that week.

2. ANONYMIZATION

All the group names and users are anonymized in this project so that no individual or group will be identified. There are 14 groups in this course. To anonymize these groups, we first came up with a list of 14 cute animal names. Then, we randomly assigned these 14 animal names to the 14 groups as aliases. We kept this assignment in a spreadsheet so that we can have a record on the mapping between the aliases and original groups. This spreadsheet will not be revealed to anyone unless required by course instructor for verification purpose.

In this report, we will only refer to groups by their aliases, for example, "group Wolf had this bad smell on ...". And we will use A, B, C, D, E to refer to a specific group member in each group, if necessary, for example, "Wolf A contributed less than ...".

Following is the list of 14 group names we will use in this report:

- Dolphin
- Rhino

- Sloth
- Hippo
- Husky
- Wolf
- Kangaroo
- Giraffe
- Zebra
- Beaver
- Eagle
- Hedgehog
- Shark
- Penguin

3. FEATURE EXTRACTION

Before starting our analysis on bad smells, we need to extract features from the raw data that we collected from GitHub repositories. We grouped the features based on the main aspect of their focus. We have features based on issues, milestones, labels, commits, and unused comments. We generated 24 features in total.

In this section, we will list by category all the features that we generated for bad smell analysis. We will also briefly explain some of the features whose meanings are not straightforward.

3.1 Issues

- Total number of issues in each group
- Number of closed issues per week
- Number of assigned issues to each person in a group
- Issue Lasting Time:
  The time between the issue's opening and closing
- Number of unclosed issues after April 7th:
  April 7th is the deadline for project 1. We counted the number of unclosed issues after project 1 deadline.
- Number and percentage of issues closed after milestone deadline
- Number and percentage of issues without milestone label
- Number and percentage of issues without labels
- Number and percentage of issues without comments
- Number and percentage of issues without assignees

3.2 Milestones

- Number of milestones closed after due date or never close
- Number of milestones with open issues after milestone closed
- Total number of milestones by group
- Number of issues in each milestone
- Number of milestones with no due date
- Number of milestone created each month

3.3 Labels

- Number of used labels:
  Include the default labels and customized labels.
- Number of unused customized labels
- Number of issues assigned to each label
- Amount of time each label lasting

3.4 Commits

- Number of commits per person/group
- Number of commits with incorrect author:
  Due to personal git setting, some commits do not have author name or the author name does not belong to any group member.

3.5 Comments

- Number of comments per person
- Number of total comments by group

4. BAD SMELL DETECTORS AND DETECTION RESULTS

After doing analysis on the features that we discussed in the previous section, we defined 16 bad smells (including 1 potential bad smell) and created detectors for each of these bad smells. We grouped the bad smells by aspects of their main focus. In this section, we will list and explain the bad smells we defined in detail. Each bad smell detector subsection will consists of three parts: justification of the bad smell, detection of the smell (method) and the result of our bad smell detector.

Before beginning listing bad smells, we will explain one common criteria we used in many of our bad smell detectors, which is the z-score. Z-score is a statistical measurement to indicate whether a value is above or below the mean value of an array by how many standard deviations. Following is the formula for z-score:

$$z - score = \frac{(x - \mu)}{\sigma}$$

In the formula, μ is the mean value of the array and σ is the standard deviation of the array. If the z-score of a value in the array is above (having positive z-score) or below (having negative z-score) the mean value over a certain threshold, this value can be considered as an outlier in the array.

In this report, we use z-score of 1.5 as our threshold. It means we consider a value is an outlier in an array if its z-score is above or below the mean by 1.5 times of standard deviation.

Besides, for each bad smell, we will assign a stink point to it based on the calculation results of corresponding bad smell detector. These stink points will be used in our ranking to select the two most "stinky" group. Details about how we used the stinky points will be explained in the Case Study section.

4.1 Issues

4.1.1 Unclosed issues after project due

Justification: A group has this bad smell if the group has unclosed issues after project due date. This bad smell is intuitive. Ideally, a group should resolve all the issues in the project before the project due. If there are issues not resolved before the deadline, it usually means the project is not finished in time. In reality, a delayed project will cost more human labor and money than what was expected.

Detection: To detect this bad smell, we looked at the "closed_time" feature of each issue and compare it with our project 1 deadline, which is April 7th, 2016. If the issue closed time is later than April 7th, we consider this issue is unfinished after project deadline. The group will score 1 stink point for having this bad smell.

Result: According to our result in Table 2, 10 out of the 14 groups suffered from this bad smell. These groups either did not succeed in resolving all the issues before project deadline, or failed to close the issue in time when the issue was resolved.

Table 2: Unclosed Issues Table

Group	Is there unclosed issues after April 7th
Dolphin	1
Rhino	1
Sloth	1
Husky	1
Wolf	1
Kangaroo	1
Giraffe	1
Zebra	1
Beaver	1
Hedgehog	1
Hippo	0
Eagle	0
Shark	0
Penguin	0

4.1.2 Issues closed after milestone deadline

Justification: A group has this bad smell if the group has issue that is closed after its corresponding milestone deadline. This bad smell is similar to the previous bad smell. The only difference is that we look at milestone deadline rather than project deadline. If an issue is not closed before the milestone deadline that it is assigned to, it means either the issue got more complicated along the way and could not be resolved in time, or the milestone deadline is set too optimistically. If not dealt carefully, this bad smell could lead to the occurrence of the bad smell we talked about in the last section.

Detection: The detection of this bad smell is also similar to the previous one. Instead of comparing issue closed time against project due date, we compare issue closed time with the milestone deadline which the issue was assigned to. If the issue closed time is later than corresponding milestone deadline, we consider the issue is not resolved before the deadline. The group will score 1 stink point for having this bad smell.

Result: Our data analysis result in Table 3 suggests that all the groups were having this bad smell. More importantly, 9 out of 14 groups have more than a quarter of issues that were closed after corresponding milestone deadline. One possible explanation is that students are bad at estimating how much time they need to solve a problem.

Table 3: Percentage of issues closed after deadline

Group	Percentage of issues closed after milestone deadline
Hippo	0.526
Shark	0.404
Husky	0.394
Eagle	0.375
Dolphin	0.371
Kangaroo	0.349
Giraffe	0.31
Rhino	0.295
Zebra	0.286
Sloth	0.221
Penguin	0.209
Hedgehog	0.2
Wolf	0.17
Beaver	0.07

4.1.3 Anomaly in opening and closing issues

Justification: A group has this bad smell if the group has significant difference in the number of opening and closing issue activities between weeks. In agile development, we expect issues are open and lose frequently throughout each cycle of developing process. This means the number of issue opening and closing event should be of approximately frequency in a given time. Here, we choose to analyze by week. Each group is expected to do same amount of work during the process.

We have total 13 weeks for this project. However, we only focus on the data from week4 to week8, and the data from week 10 to week 13. Because the data shows the in the first three weeks (before Feb 1st) and spring break of week 9, only few groups had open and closed issues. The details of this data is shown in Table 21 in section 5.1.2.

Detection: The detection of this bad smell uses z-score. For each single group, we first get the combined number of opening and closing issue events by week. This will give us an array that looks like [2,3,2,1,5,.....]. The length of this

array should be 12, because the length of project 1 is 13 weeks and we removed the week for spring break. Next, we calculate a z-score for the group based on the array. If one of the z-score is greater than 1.5 or less than -1.5, we consider that the group had anomaly in the frequency of opening and closing issue events. Corresponding group will gain 1 stink point for having this bad smell.

Result: Our data analysis in Table 4 shows that all groups have this bad smell. It means all groups were not generating and resolving issues steadily through the semester. This is probably because all the group pushed two much before the deadline.

Table 4: Abnormal open or close issues activity per week

Group	Abnormal open or close issues activity per week
Dolphin	1
Sloth	1
Hippo	1
Husky	1
Wolf	1
Kangaroo	1
Giraffe	1
Zebra	1
Beaver	1
Eagle	1
Hedgehog	1
Rhino	1
Shark	1
Penguin	1

4.1.4 Issues without milestone

Justification: A group has this bad smell if the group has an issue that is not assigned to any milestone. For an agile project, all issues are expected to be solved sometime before project deadline. However, each issue itself does not have a deadline. The best way to set a deadline to an issue is by assigning it to a milestone, indicating the issue needs to be resolved before the deadline of the milestone. Failing to assign an issue to a milestone might create confusion on before when should the issue be resolved and resulting in having open issue after the project is finished. Thus, we expect all the issues are assigned to a milestone.

Detection: To detect this bad smell, we simply get all the issues created by each group and count the number of issues that are not assigned to any milestone. Then, we use the percentage of issues that are not assigned to any milestone to give stink points to each group. For example, if a group has 4 out of 10 issues created that are not assigned to any milestone, the group will receive 0.4 stink point for this bad smell.

Result: Our data analysis in Table 5 shows that 12 out of the 14 groups has this bad smell. Among all the groups, half of them has a quarter of issues that weren't assigned a milestone.

4.1.5 Issues without labels

Table 5: Percentage of Issues Without Milestone

Group	Percentage of issues without milestone label
Beaver	0.825
Sloth	0.735
Rhino	0.695
Wolf	0.547
Hippo	0.342
Dolphin	0.286
Giraffe	0.253
Husky	0.212
Eagle	0.143
Zebra	0.143
Hedgehog	0.067
Shark	0.038
Penguin	0
Kangaroo	0

Justification: A group has this bad smell if the group has an issue that has no label assigned to it. Labels serves a good way to organize issues. Assigning meaningful labels to issues can save contributor time and effort to find a single issue or a category of issues. It is a good practice to always assign at least 1 label to any issue.

Detection: To detect this bad smell, we simply get all the issues created by each group and count the number of issues with no labels. Then, we use the percentage of issues that has no label to give stink points to each group.

Result: Our data analysis in Table 6 shows that 13 out of the 14 groups has this bad smell and 6 out of all the groups has more than a quarter of issues without labels. It is surprising that so many groups are having this problem because instructor explicitly said to have at least 1 or 2 labels for each issue.

Table 6: Percentage of Issues Without Labels

Group	Percentage of issues without any label
Hippo	0.658
Sloth	0.529
Wolf	0.528
Rhino	0.442
Husky	0.333
Beaver	0.281
Dolphin	0.200
Giraffe	0.195
Eagle	0.161
Shark	0.154
Zebra	0.143
Hedgehog	0.100
Kangaroo	0.035
Penguin	0

4.1.6 Issues without comments

Justification: A group has this bad smell if the group has an issue that has no comments. The comment area serves

both as a platform for communication and a place to record progress on the issue. Contributors can put down their questions in commenting area and assign someone else in the group to answer. It is a good practice to leave a comment before closing an issue stating how the issue is resolved. If the issue is resolved by a commit, it is also good to include the SHA code for the commit in the commenting area before closing the issue, so that other people can trace back to the specific commit in the future. Thus, we think all the issues should have at least 1 comment.

Detection: To detect this bad smell, we simply get all the issues created by each group and count the number of issues that don't have any comments. Then, we use the percentage of issues that has no label to give stink points to each group. For example, if a group has 4 out of 10 issues created that have no comments, the group will receive 0.4 stink point for this bad smell.

One thing worth mention is how we dealt with unclosed issues after project due. Normally, it is understandable that a newly open issue does not have any comments. People might not be ready to solve this issue. These issues should not be counted into bad smells. However, no group has more than 10% of unclosed issues at the end of the project. These uncommented, unclosed issue won't contribute too much to the stink point. Thus, we spared the effort to clean them out.

Result: Our data analysis in Table 7 shows that all of the 14 groups had this bad smell and as many as 10 of them has more than a quarter issues that have no comments. It's probably because it's first exposure to collaborate on GitHub for many of the students and they're not used to use virtual platform as a hub for discussion instead of having face to face meeting or communicating via email.

Table 7: Percentage of Issues Without Comments

Group	Percentage of issues without comments
Rhino	0.695
Kangaroo	0.616
Beaver	0.614
Dolphin	0.514
Sloth	0.456
Zebra	0.347
Penguin	0.326
Wolf	0.302
Hedgehog	0.278
Eagle	0.268
Husky	0.242
Giraffe	0.218
Shark	0.212
Hippo	0.158

4.1.7 Issues without assignee

Justification: A group has this bad smell if the group has an issue that does not have assignees. A good practice when creating an issue is to assign a person to be responsible for resolving this issue. This will make sure that all the issue has at least one person who's taking charge of. Sometimes, multiple people are working on a same issue, but GitHub only allow one assignee for each issue. In this situation, one people should be selected to take lead on resolving and updating the status of this issue. Even if it is not certain who should be responsible for the issue at the moment the issue is created, it should be assigned to a person so that this person can remember to reassign the issue to other persons when determined. Thus, any issue that does not have an assignee is a bad smell.

Detection: To detect this bad smell, we simply get all the issues created by each group and count the number of issues that does not have an assignee. Then, we use the percentage of issues that has no assignee to give stink points to each group. For example, if a group has 4 out of 10 issues created that have no assignee, the group will receive 0.4 stink point for this bad smell.

Result: Our data analysis in Table 8 shows that 13 out of the 14 groups had this bad smell and 9 of the 14 groups had more than a quarter of issues without assignee. According to our group's experience, we assume this happens most when people think they should collaborate on solving the issue but they couldn't set more than 1 assignee to the issue. A recommended practice would be electing a leading role for each issue and assign that person to the issue to make sure every issue is taken care of.

Table 8: Percentage of Issues Without Assignee

Group	Percentage of issues without assignee
Giraffe	0.782
Sloth	0.735
Dolphin	0.686
Wolf	0.509
Rhino	0.505
Husky	0.455
Eagle	0.429
Beaver	0.404
Hippo	0.316
Zebra	0.143
Shark	0.115
Hedgehog	0.100
Penguin	0.093
Kangaroo	0

4.1.8 Abnormally Long or Short Issue Length

Justification: A group has more of this bad smell, if they have more abnormally short or abnormally long issues. It's a good practice to split the jobs into small tasks almost equally hard to assign to people. We think having very short or very long issues is a bad smell because for the long issues it can show not being specific in the task definition and including different tasks in one issue, and for short issues it might show that they just opened and closed the issue to show that they used issues, not actually for tracking and solving it.

Detection: To detect this bad smell, we calculated z-score for each issue length and counted the ones more than 1.5 or less than -1.5 and considered them to be outliers.

Then we found the percentage of those outliers and recorded it as a bad smell score.

Result: Our data analysis in Table 9 shows that 5 groups have more than 10% unusual issue length.

Table 9: Percentage of Issues with Unusual Length

Group	percentage of Issues with unusually long or short lasting time
Husky	0.125
Eagle	0.107
Sloth	0.106
Zebra	0.102
Hedgehog	0.100
Shark	0.096
Rhino	0.095
Giraffe	0.092
Beaver	0.088
Dolphin	0.086
Kangaroo	0.082
Wolf	0.075
Hippo	0.026
Penguin	0.023

4.2 Labels

4.2.1 Unused label

Justification: A group has this bad smell if it has self-created, unused label in its repository. Labels serve a good way to organize issues. GitHub has given a set of default labels for user to use. It also allows users to create customized labels based on their needs. It is good to have a variety of labels for different kinds of issues. However, sometimes people might have customized labels that are never used in the project. These labels has seemingly reasonable names. However, they're either redundant labels that have the same meaning with other labels, or they're originally planned to be used but no issue is a good match with the labels. Having unused label in the repository may harm the organization of issues in that it creates confusion and inconvenience when people are looking for a single or a category of issue.

Detection: Detection of this bad smell is straightforward. For each group, first, we get a list of labels in its repository. Next, we clean out the label information from GitHub default labels, because default labels are general labels that do not have strong description power. Many of the default labels are not used by most of the groups. We only consider self-created labels for this bad smell. Then, we calculate the number of issues assigned to each of the labels. If the group has a customized label that no issue is assigned to, we consider the group has this bad smell. This group will receive 1 stink point for this.

Result: Our data analysis in Table 10 shows that only 2 out of the 14 groups had this bad smell. This means most of the groups used at least 1 time of each label they created.

4.3 Comments

4.3.1 Outlier in number of comments

Table 10: Having Unused Custom Labels

Group	Has unused custom labels
Rhino	1
Sloth	1
Dolphin	0
Hippo	0
Husky	0
Wolf	0
Kangaroo	0
Giraffe	0
Zebra	0
Beaver	0
Eagle	0
Hedgehog	0
Shark	0
Penguin	0

Justification: A group has this bad smell if the group has significant difference between the number of comments made by each group member. As one form of contribution to the project, we expect every member should have approximately the same amount of comments in all the issues. This is also a way to see if GitHub commenting area served as a good communication platform for the group. If there are some group members who rarely comment on GitHub and other group members comment a lot, it means the group did not have good coordination on the use of GitHub issue. Thus, we consider this a bad smell.

Detection: To detect this bad smell, we use z-score. For each single group, we get the number of comments made by each group member and save the numbers into an array. The array will look like [15, 19, 20, 13]. The length of the array is the same as the number of group members. Then, we calculate z-score based on this array. If a group member's z-score is greater than 1.5 or less than -1.5, we consider this group has this bad smell. The group will receive 1 stink point for having this bad smell.

Result: Our data analysis in Table 11 shows that 9 out of the 14 groups had this bad smell. It means a little more than half of the groups have uneven communications in comments.

4.4 Commits

4.4.1 Outlier in group commit

Justification: A group has this bad smell if there is significant difference between the number of commits made by each group member. The number of commits is an very important indicator on how much contribution a group member has made on a project. Ideally, we expect every group member to make roughly the same amount of contribution to the project.

A common practice for making commits is to commit as often as possible when the code is changed. This will make it easier to track the change of code and also easier to restore the latest version if user wants to discard recent changes. People may have different habits on how much work needs to be done before making a commit. However, if a member

Table 11: Users Not Participating in Issue Comments

Group	Has outlier user in number of comments
Rhino	1
Sloth	1
Dolphin	1
Hippo	1
Eagle	1
Shark	1
Kangaroo	1
Husky	1
Penguin	1
Zebra	0
Wolf	0
Beaver	0
Hedgehog	0
Giraffe	0

Table 12: People Not Participating Equally in Group Commits

Group	Has outlier in group commits
Zebra	1
Shark	1
Eagle	1
Hippo	0
Dolphin	0
Sloth	0
Beaver	0
Penguin	0
Giraffe	0
Hedgehog	0
Kangaroo	0
Husky	0
Wolf	0
Rhino	0

have significantly more or less commits than other group members, it either means the group member has made significantly more or less contribution to the project, or the group members did not have agreement on how much work should be done before making a commit. In both cases, we consider the group is having this bad smell.

Detection: The detection of this bad smell uses z-score. For each single group, we first get the total number of commits of each group member for project 1 and store these numbers in an array. This will give us an array that looks like [35,28,43,13]. The length of the array is the same with the number of group members. Next, we calculate z-scores for the group based on the array. If any z-score is greater than 1.5 or less than -1.5, we consider that the group had this bad smell. Corresponding group will gain 1 stink point for having this bad smell.

Result: Our data analysis in Table 12 shows that 3 out of the 14 groups has this bad smell. It means the majority of groups have good task distribution, each group member has similar workload.

4.4.2 Total commits

Justification: A group has this bad smell if the average number of commits in the group has significant difference with all other groups. This bad smell is more domestic for this course project. As a course project, we expect the average workload for every participants in class to be the same. Here we made an assumption that everyone has the approximately the same amount of work committed in each commit. Based on this assumption, if the average number of commits of a group is significantly less than that of other groups, it means the group did not do enough work as expected for the number of members in the group. This is a bad smell. If the average number of commits of a group is significantly higher than that of other groups, on the other hand, it might mean that the group really did significantly more work than other groups. However, it might also mean that the group committed less work in each of their commits and this is bad smell. In this case, we will also report this as a bad smell to notify the anomaly. Whether it is really a bad smell will need other ways to make the judgement.

Detection: To detect this bad smell, we simply get all the total commits created by each group and count the *commits/person* for each group. Then, we record the maximum *commits/person* and calculate the inverse percentage of the commits/person based on the maximum *commits/person* for all groups. We treat those groups whose *commit/person* value is less than 75% of the best group as they have bad smells.

Result: Our data analysis in Table 13 shows that 11 out of the 14 groups had this bad smell.

Table 13: Having less Average Commit per Person Compared to Max in All Groups

Group	Low average commits per person compared to Max group
Husky	0.755
Shark	0.726
Eagle	0.513
Wolf	0.502
Beaver	0.453
Dolphin	0.436
Penguin	0.360
Rhino	0.329
Hippo	0.329
Zebra	0.307
Giraffe	0.253
Kangaroo	0.136
Hedgehog	0.120
Sloth	0

4.5 Milestone

4.5.1 Bad milestone frequency

Justification: A group has this bad smell if the group has significantly different frequency of milestone activity, specifically, opening and closing milestone. Agile development is

about regularly delivering usable software during development cycle. In agile, the development process is expected to be decomposed into multiple, roughly equal length, short sprints. Each of the short sprint has a milestone for the current developing period. At the end of each sprint, a periodical goal should be achieved. In this sense, we expect milestone to be open and closed frequently and regularly throughout the development of project 1. If milestones opening and closing do not have a similar frequency in project 1, it might indicate that instead of agile, the group used waterfall developing process, which creates all the milestones at the beginning of the project, or the group did not consistently work on the project. Either way, it is a bad smell and the group that has this bad smell will get 1 stink point for it.

Detection: The detection of this bad smell uses z-score as before. For each single group, we first get the combined number of opening and closing issue events by month. This will give us an array that looks like [2,3,2,1]. Next, we calculate a z-score for the group based on the array. If the z-score is greater than 1.5 or less than -1.5, we consider that the group had this bad smell. Corresponding group will gain 1 stink point for having this bad smell.

Result: Our data analysis in Table 14 shows that only 2 out of the 14 groups has this bad smell, meaning most of the groups are doing well on avoiding this bad smell.

Table 14: Abnormal Milestone Activity in All Groups

Group	Abnormal Milestone Activity
Zebra	1
Wolf	1
Hippo	0
Dolphin	0
Sloth	0
Beaver	0
Penguin	0
Giraffe	0
Hedgehog	0
Shark	0
Kangaroo	0
Husky	0
Eagle	0
Rhino	0

4.5.2 Abnormal Issue number in milestone

Justification: A group has this bad smell if the group has unexpectedly large or small number of issues in a single milestone. As we talked about in last section, we expect milestones to be opened and closed with consistent frequency though project 1. It means each milestone should last roughly the same amount of time in order to maintain the frequency. This indicates that each milestone should have approximately the same number of issues within it. If a group has a milestone A that consists of 2 issues and another milestone B that consists of 8 issues, it might means the issues in milestone A are not breakdown good enough or the workload of the two milestones are very unbalance, probably results from inconsistency of effort through the project. In both cases, it is a bad smell. Group that has this bad smell will get 1 stink point.

Detection: Like last bad smell, we also use z-score to detect this bad smell. For each group, we summarize the number of issues assigned to each milestone and put the numbers into an array. The array will look like this: [5,8,6,3,4]. The length of the array is the same as the number of milestones created by the group. After getting the array for the group, we calculate a z-score based on this array. If the z-score is greater than 1.5 or less than -1.5, we consider that the group had this bad smell. This group will gain 1 stink point for having this bad smell.

Result: Our data analysis in Table 15 shows that 11 out of the 14 groups had this bad smell. However, none of them has over even 15% of the milestones that has abnormal number of issues assigned to them.

Table 15: Abnormal Issue Number in Milestone

Group	total number of milestone	number of milestone with abnormal issue number	percentage of milestone with abnormal issue number
Penguin	7	1	0.143
Kangaroo	7	1	0.143
Beaver	7	1	0.143
Hedgehog	9	1	0.111
Shark	9	1	0.111
Sloth	11	1	0.100
Giraffe	10	1	0.100
Dolphin	12	1	0.083
Wolf	16	1	0.063
Hippo	17	1	0.059
Zebra	19	1	0.053
Husky	4	0	0.000
Eagle	12	0	0.000
Rhino	7	0	0.000

4.5.3 Missing due date

Justification: A group has this bad smell if the group has a milestone that has no deadline. A milestone is called a milestone because it sets a periodic goal for the project. All milestone needs to be finished in a given time. A milestone will be meaningless after the project is already finished. In this sense, the latest due date for a milestone should be the same with project deadline, but never without a due date. A missing due date may cause the delay of the whole project.

Detection: For detecting this bad smell, we just see if the "due_date" of each milestone exists. If a milestone does not have a due date, the group will receive 1 stink point for having this bad smell.

Result: Our data analysis in Table 16 shows that 11 out of the 14 groups had this bad smell. Even though only 2 of the 14 groups had slightly more than a quarter of milestones that have no due date, we still think not a single milestone should be left without a due date. Use the date the milestone needs to be updated if the due date is not certain.

Table 16: Percentage of Milestones with no Due Time

Group	Percentage of Milestones with no Due Time
Kangaroo	0.286
Zebra	0.263
Sloth	0.200
Wolf	0.188
Dolphin	0.167
Beaver	0.143
Penguin	0.143
Hippo	0.118
Shark	0.111
Hedgehog	0.111
Eagle	0.083
Rhino	0.000
Husky	0.000
Giraffe	0.000

Table 17: Groups Having Less Total Per Person Comments

Group	low average comments per person compared to Max average group
Beaver	0.868
Kangaroo	0.845
Rhino	0.816
Zebra	0.802
Dolphin	0.756
Shark	0.706
Eagle	0.692
Wolf	0.691
Husky	0.675
Hippo	0.631
Penguin	0.607
Hedgehog	0.484
Sloth	0.292
Giraffe	0

4.6 Potential Bad smells

4.6.1 Average comments per person

Justification: A group has this bad smell if the average number of comments made in the group has significant difference with that of other groups. This is a course related bad smell. Ideally, we expect all the participants in class has approximately the same amount of communication (mainly in the form of making comments to issues) on GitHub. If the average number of comments made within a group has significant difference with that in other groups, it means the group is having communication problems compared to other groups. This is almost no doubt a bad smell. However, this bad smell needs to be based on a rather strong assumption that all the groups created roughly the same amount of issues. If a group has less issues created, then it is more likely that the group has less comments made. However, the bad smell for this lies in number of issues created rather than number of comments made. This is the reason why we consider it as a potential bad smell.

Detection: The detection of this bad smell uses z-score. For each single group, we first calculate the average number of comments made by each group member by dividing the total number of comments made by the group with total number of group members. Then, we store the average number of comments for all the groups into an array. This will give us an array that looks like [35,28,43,13,...]. The length of the array should be the same as the number of groups in this course, which is 14. Next, we calculate z-scores for all the groups based on the array. If any z-score is greater than 1.5 or less than -1.5, we consider that the corresponding group had this bad smell. Corresponding group will gain 1 stink point for having this bad smell.

Result: Our data analysis in Table 17 shows that 13 out of the 14 groups have this bad smell.

5. CASE STUDY

After defining the stink points for each bad smell that we found, we added all the stink points together by group, and further, get a ranking of total stink points for all groups. The overall ranking is shown in Table 18. Then, we selected the two groups with highest stink points, along with our own group, for our case study. We will use these three groups, Rhino, Dolphin, and Hedgehog, as exemplars to demonstrate the existence of the bad smells we discussed in section 4.

Table 18: Final Group Rankings Based on Stink Point

Group Name	Final Stink Point
Rhino	8.87
Dolphin	7.59
Zebra	7.59
Sloth	7.38
Eagle	6.77
Shark	6.67
Wolf	6.58
Kangaroo	6.49
Husky	6.19
Hippo	6.16
Beaver	5.89
Giraffe	5.20
Penguin	4.90
Hedgehog	4.67

5.1 Issues

Table 19 shows these three groups' issue activities (opening and closing of issue). Taking a close look at the table, in the first row, group Dolphin has a total number of 35 issue activities, which is much less compared to the other two groups. This is a bad smell because the number of issues open or closed is an indicator of the amount of problems the group found and resolved for the project. Having lower number of issues shows either they didn't do much work, or they didn't open and close enough issues for the work they did.

The second row of Table 19, "Abnormal length", indicates the number of issues with abnormal short or long lasting

time. We consider both conditions are bad, because abnormal short time, in extreme cases the issue lasted for only a few seconds, shows that they have just opened and closed the issue to save the work that they've already done, and not actually to record work or show progress or make communication. The abnormal long issue lasting time may indicate that they did not divide their tasks small enough to be able to solve each one with in small amount of time. The data show that all these groups had about 10% abnormally short or long issues.

The third row in the table is issue closed after April 9th, which indicates either the group didn't finished work on time, or they forgot to close them. The result shows that all of these groups have at least one issue remain open after April 9th.

Next, we discuss about the number of issues that have been closed after milestone deadline. This indicates late delivery of work and it is a bad smell. It is either because their work were not done on time or they closed all the issues and the milestones at once, not at the deadline or the meeting, but just when they have time. We see these groups have quite many of this type of issues, especially for Group Rhino, which has as many as 28.

At last, we show issue counts with no comment, no milestone, no label and no assignee. This is a bad smell because it shows not using GitHub issues properly. All these three groups have had these issues. Group Rhino has significantly more occurence all of these bad smells than the other two groups.

Table 19: Number of Issues

Item	Dolphin	Rhino	Hedgehog
Total	35	95	90
Abnormal Length	3	9	9
Close after Apr.9	3	1	1
Close after Milestone	13	28	18
No Comments	18	66	25
No Milestone	10	66	6
No Label	7	42	9
No Assignees	24	48	9

5.1.1 Assignee

The assignee related data is shown in Table 20. We used the data to find if there is any person who was assigned to too many or too few issues. From Table 20 we can see that issue assignment is pretty even in group Dolphin and group Rhino. However, in group Hedgehog, User-B was assigned to many more issues than the others. Either the other group members didn't do as much work than User-B, or User-B led many of the discussion on resolving many of the issues.

Table 20: Issues Assignees

Item	Dolphin	Rhino	Hedgehog
User-A	4	13	15
User-B	2	13	37
User-C	3	18	11
User-D	2	13	18

5.1.2 Open and Close Issue Activities

Table 21 shows the number of opened and closed issues in each week for each group that starts from Jan-09, 2016 to April -07, 2016, 13 weeks in total. We excluded the first three weeks (not highlighted in the table) since that's when coding hadn't started yet. Also, week 9 is excluded because it was the spring break and no group had any issue opening and closing activity. From this table we can clearly see that for group Rhino, they had significant more issue opening and closing activities than any other weeks combined. This is exactly the kind of bad smell we were trying to detect.

Table 22 shows abnormal weeks in opening and closing issues. Generally speaking, all three groups have abnormal weekly activities in week 7 or week 8, which was very close to our March 1st project deadline. It seems like all the group were pushing too much before the deadline. This is a bad smell, because it shows they piled up their work instead of doing it gradually.

Table 22: Number of Weeks Involved with Abnormal Open and Close Issues Activities

Activity	Dolphin	Rhino	Hedgehog
Weekly open issues	W7&W8	W8	W7
Weekly close issues	W7&W8	W8	W7

5.1.3 Labels

The total number of unused customized labels for the three groups are shown in Table 23. We can see that group Rhino is the only group among the three that has unused costomized label.

Tables 24, 25, and 26 show each group's label names with their total used count and average lasting length in issues. From these tables, we can see that these three groups had very different customized labels. On a closer look, we see that some of the labels were used a lot more times than others, indicating they had more issues related to these labels. For example, group Dolphin has most of its issues in data-collection, Hedgehog group has most of them in Feature and Rhino group has the most in UI. Also looking at the average time in these issues can be interesting, because it shows which label's tasks took more time.

Table 21: Number of Issues Opened and Closed per Week

		1/16 W1	2/23 W2	1/30 W3	2/6 W4	2/13 W5	2/20 W6	2/27 W7	3/5 W8	3/12 W9	3/19 W10	3/16 W11	4/2 W12	4/9 W13
Dolphin	Open	0	0	0	1	6	2	12	11	0	0	0	0	0
	Close	0	0	0	0	0	4	12	15	0	0	0	1	0
Rhino	Open	0	0	2	1	0	0	15	69	0	0	0	0	7
	Close	0	0	0	0	2	0	10	75	0	0	0	0	7
Hedgehog	Open	1	6	2	5	0	14	22	16	0	8	12	3	0
	Close	1	0	1	7	0	7	23	19	0	7	13	11	0

Table 23: Labels

Item	Dolphin	Rhino	Hedgehog
Total	7	12	14
Unused-customized	0	4	0

Table 24: Average Length and Total Number of Issues in each Label, Group Dolphin

Lable Name	Count	Average Time(Hour)
data-collection	11	109.20
enhancement	1	210.71
front-end	4	11.67
help wanted	2	367.41
plugin-development	3	145.33
proj1 feedback	1	367.80
Text mining	6	210.44

Table 25: Average Length and Total Number of Issues in each Label, Group Hedgehog

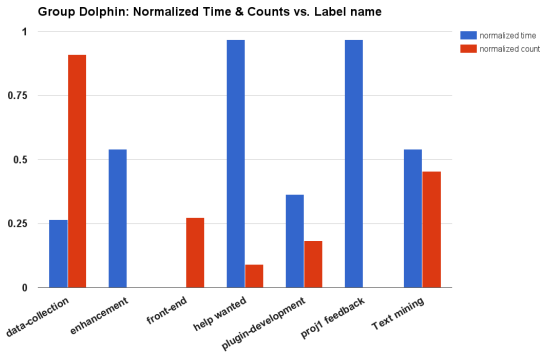
Lable Name	Counts	Average Time(Hour)
bug	9	138.25
design	20	186.75
enhancement	15	71.73
Evalutation	6	248.80
feature	33	180.92
help wanted	15	123.91
in progress	50	169.36
in testing	5	114.68
optimize	2	105.12
question	3	138.71
Solution One	27	104.77
Solution Three	32	181.35
Solution Two	19	174.40
wontfix	1	14.08

Table 26: Average Length and Total Number of Issues in each Label, Group Rhino

Lable Name	Counts	Average Time(Hour)
API	10	16.24
bug	3	0.57
database	1	2.01
design	9	86.758
Documentation	4	24.20
enhancement	1	172.92
Integration	2	22.53
Notification/Calendar solution	1	244.60
PDF solution	1	21.19
ProjectPlanning	5	182.66
solutions	2	209.21
UI	16	37.924

We also see a graph of normalized number of issues and issue lasting time for the three groups in figures 7, 8, and 9 that can show us relatively which ones took more time or more issues. They are normalized between 0 and 1 so we can compare them among groups for common labels, too. For example, we can see that group Dolphin has put more issues in enhancement than the other two, but then group Hedgehog had more issues in design.

Figure 7: Group Dolphin: Labels usage counts and lasting time



5.2 Comments

We counted the number of comments each member has made on issues and tried to find outliers within these numbers.

Figure 8: Group Hedgehog: Labels usage counts and lasting time

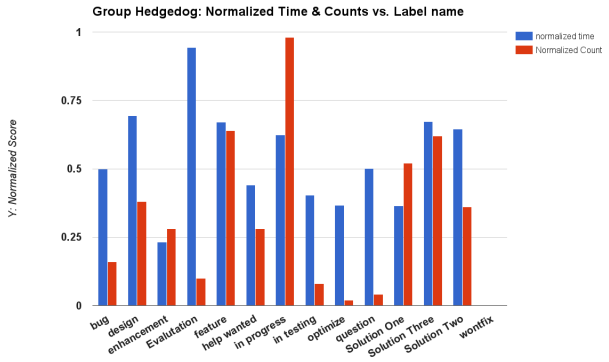
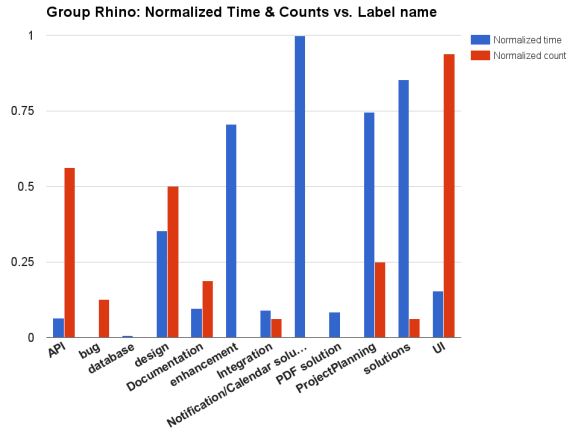


Figure 9: Group Rhino: Labels usage counts and lasting time



We believe that having members who do not participate in comments evenly can mean that the communication was not good within the group. From Table 5.2, we can see that in group Rhino, Rhino A did not make any comments on any issue. On the other hand, in group Hedgehog, Hedgehog C made many more comments than the others. We consider both cases bad smell.

5.3 Commits

We calculated the commits each person made in the three groups. The summarized stats is shown in table 5.3. From table 5.3, both group Dolphin and Rhino don't have equal contribution among users. Specifically, Dolphin D made the most contribution in the whole group, while Dolphin A and Dolphin B did almost nothing. The same thing happened between Rhino D, Rhino C and Rhino A as well, where Rhino D and Rhino C made the most contribution within the team and Rhino A did almost nothing. We also draw a weekly commits graph (Figure 10, Figure 11 and Figure 12) to show the huge difference between the number of commits for each group member..

Table 27: Total Number of Comments per Person

Contributor	Dolphin	Rhino	Hedgehog
A	3	0	11
B	13	13	38
C	3	19	62
D	6	19	25

Table 28: Total Number of Commits per Person

User	Dolphin	Rhino	Hedgehog
A	2	1	29
B	9	13	33
C	28	28	43
D	78	31	46

Those figures(10,12,11) show that both Group Dolphin and Group Rhino have two main contributors among four users. For Group Rhino, they always catch up the deadline and make massive commits in a short time.

5.4 Milestones

Since milestones show the goals of each group, we calculate the number of milestones of each group monthly. The results of the milestone stats are shown in table 5.4.

We use our milestone bad smell detector to detect 1.abnormal issue numbers in milestones, 2.no due time on milestones and 3.abnormal milestone activity. The result of those three groups are shown in table 5.4.

From table 5.4, both Group Dolphin and Hedgehog have some bad smells for the milestones. Group Dolphin has one milestone with abnormal issues and two milestones without due time.

6. CONCLUSION

In this report, we analyzed project 1 Github usage data from all 14 groups in CSC510 Software Engineering course. We extracted 24 features from all the data collected from

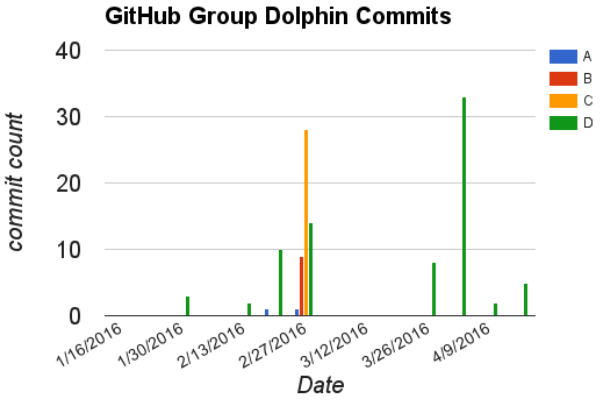


Figure 10: Group Dolphin Weekly Commits

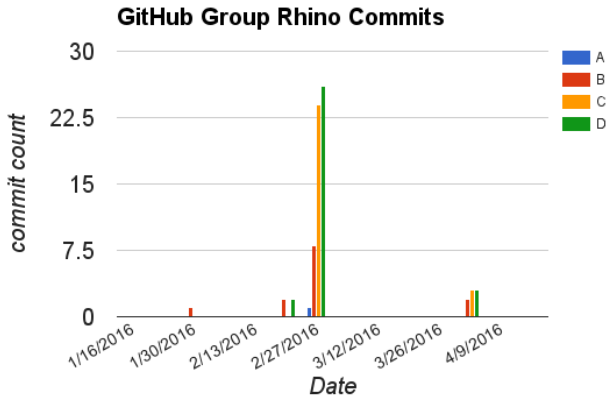


Figure 11: Group Rhino Weekly Commits

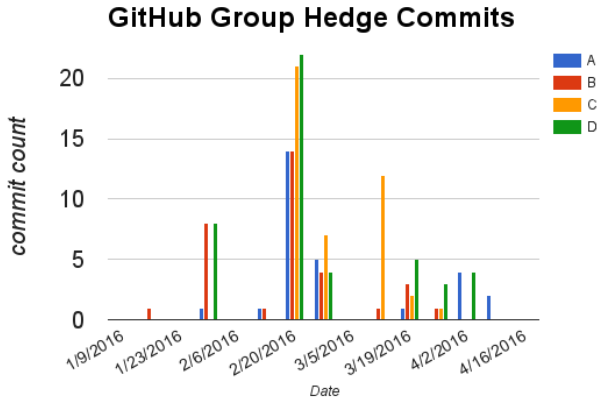


Figure 12: Group Hedgehog Weekly Commits

each project repository. After close inspection of these features, we abstracted 16 bad smells that occurred within all the groups. Then, we discussed in detail about each bad smell we found by making justification for the bad smell, explaining detection method for the bad smell, illustrating and discussing our detection result of the bad smell.

Based on our definition of bad smells, we conclude that there is no perfect group in terms of Github usage in project 1. We generalize the common problems that most groups suffered from as below:

- In regard to the issue usage, many groups were having significantly more issue activities before the deadline of March 1st than that of other times, which is considered not good.
- Some groups didn't have any issues opened or closed after March 1st deadline. Since we were applying agile development process in our projects, there should be continuous issue activities after March 1st deadline to keep on testing and improving our projects.
- It appears that all the groups had problems with appropriately labeling issues, as well as opening and closing issues and milestones on time.

Table 29: Number of Monthly Milestone

Time	Dolphin	Rhino	Hedgehog
January	0	1	4
February	8	4	4
March	0	0	1
April	4	2	0

Table 30: Number of Monthly Milestone

Group	Total mile-stones	Milestones without due time	Abnormal Issue Numbers
Dolphin	12	2	1
Rhino	7	0	0
Hedgehog	9	1	1

We also make the following suggestions to collaborations on project using Github:

- Plan ahead. Try to split workload more evenly throughout the process to ensure the quality of the project. Use milestone appropriately to help plan.
- Clear task distribution. Create issues with suitable milestones and labels. Assign issues to certain group members. If an issue requires collaboration to resolve, assign the person who takes charge of resolving the issue.
- Stick to the plan. Try to resolve all issues within their milestone deadlines. Procrastination will make workload distribute unevenly and causes threats to project finishing time and quality.

7. EARLY DETECTOR

Our bad smell detector shows that there is no perfect group. Even the group with lowest stinky points has some bad smells. Bad smells may have negative effects on project progress. Hence it's good to identify and avoid those bad smells as early as possible.

Here, we propose our early bad smell detectors. The intuition for our early detector is that when people procrastinate and catch up the deadline under pressure, they are more likely to make mistakes than those who have good schedules. Catching up deadline in a short time may cause many bad smells including opening massive issues within a short time, opening issues without labels and assignees, and closing issues after milestone deadline, etc. Our early detector can predict whether a group will have those bad smells in future by comparing its progress to other groups. If a group is falling behind, it is more likely to have bad smells in the future. We will give this group warning before bad smell happens.

7.1 Early Detector Model

We build our detector model based on following assumptions:

- Every group has the same deadline for their projects.
- Each group makes progress according to issues and milestones.
- Each group should make some progress on their projects per half-month.
- Catching up deadline in a short time will cause bad smells.

When people procrastinate and catch up the deadline under pressure, they are more likely to make mistakes than those who have good schedules. Our model will first, compare the commits, comments and issues between all 14 groups at each week to see if there is any group that is falling behind the others. We calculate the z-score about those features for each group and filter those groups who have more than two successive lower z-score (less than -0.5). We give warning to these groups that fall behind in the current state and my have bad smells in future. The scripts for the early detector model are included in our GitHub repository.

7.2 Early Detector Results

In this section, we will show our the early detection results for the same three groups as we specified in our Case Study section. Figure 13, 14, 15 show the result for group Dolphin, Rhino and Hedgehog respectively. We highlighted the bad smell happening time as red dot in graphs according to Table22 and the warning time as yellow flag. The time is split by weeks. There are 10 weeks in total. We ignore the first three weeks from Jan, 9 to Jan, 30 and Spring Break data, since almost all the groups didn't start to coding and there are no commits in these weeks. Our model can detect the bad smells ahead automatically. Here we just use issues closed in each week as an example. Our early detector can also be applied to opened issues, weekly commits and comments.

Figure 13 shows the results for Group Dolphin, our early detector will give them warning notification in week 5, week 10, week 11, week 12 and week 13.

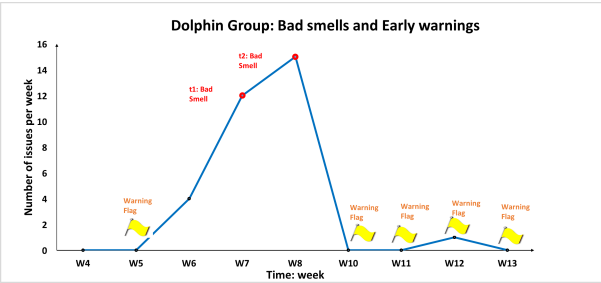


Figure 13: Dolphin: Bad smells and early warnings

Group Rhino result is shown in Figure 14, we give them warning notification in week 6, week 11, week 12 and week 13.

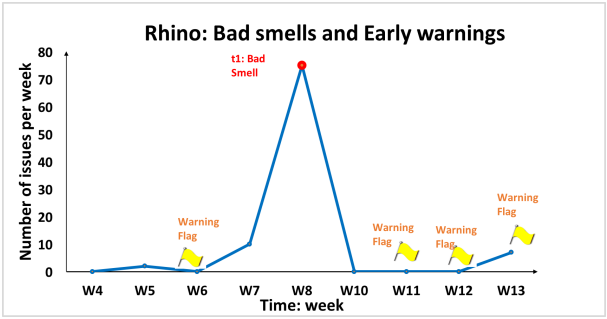


Figure 14: Rhino: Bad smells and early warnings

For Group Hedgehog in Figure 15, we give them warning notification in week 13. Our model failed to give Group Hedgehog warnings before the bad smell happens. Because our model takes previous efforts into consideration, our model give each group one week to catch up on their progress. Since group Hedgehog has a good amount of closed issues in week 4 and week 6, our model treats week 5 as in progress, so it didn't give early waring at week 5.

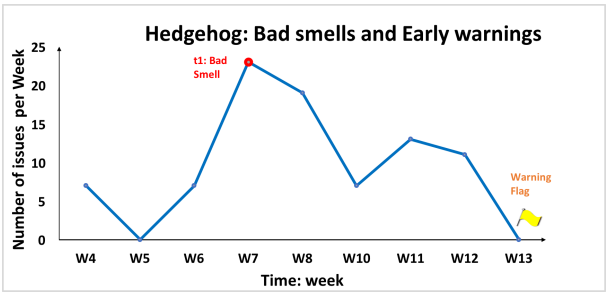


Figure 15: Hedgehog: Bad smells and early warnings

The results show that our early bad smell detector can predict the bad smell well for group Rhino and group Dolphin, which are the two groups with highest stink score. Two major bad smells of group Rhino and group Dolphin are detected two weeks before it happens. Our early detector also gives warnings in week 11-13 to both Groups, Dolphin and Rhino, but there is no bad smell afterwards. We give this warning based on our assumptions and this warning show that they need to put more effort on making progress compared with other groups. Otherwise bad smell probably will happen after these weeks. Our model fails to give warnings before the bad smells of Hedgehog happen. To deal with this situation, we will need some more strong assumptions, like each group must make some progress on their project per week instead of per half-month.