Informedb Software Quality Metrics

Liam Whitenack

2021-06-04

This project was done as an Internship Report to be presented at the end of the year. It is made in an attempt to find the eight Software Quality Metrics found in the PowerPoint attached.

All data is collected from the Informedb project. The project has assigned issue resolution data on Jira and Has commit data on GitLab that was collected and organized to analyze in RStudio.

Question of Interest

The issue I will be addressing will be the military's request for organized and presented data on projects. Software Quality Metrics are useful for analyzing trends and discovering patterns that lead to productivity. This project is being developed because the military would like reports on these metrics but doing some data analysis could show how the company can improve the efficiency of their software development.
I will be addressing the issue by mining all the data available on Jira and GitHub for the Informedb Project on the company's progress in finding issues, starting work on said issues, and resolving the issues. The project will require me to find eight different Software Quality Metrics:

- Direct Trends
- Total Defects found in the last 4 weeks
- MTTR – Mean time to repair
- Direct Removal Efficiency
- Number of failed fix attempts (external)
- Number of failed fix attempts (internal)
- Change failure rate within release cycle
- Automated Code Coverage (in percent)

```
GitLab <- GitLab %>%
  mutate(
    author_date = as.Date(strptime(author_date,
                                   format = "%a %b %d %H:%M:%S %Y")),
    committer_date = as.Date(strptime(committer_date,
                                      format = "%a %b %d %H:%M:%S %Y"))
    # read columns as dates instead of characters
  )

GitLab_reduced <- GitLab %>%
```

```
  select(
    author_date,
    committer_date
  )
```

This code block takes all of the data from GitLab and makes it usable. It then outputs it all into a much simpler dataset which only contains the columns being used.

```
for(i in 1:length(Jira$Status)){
  Jira$Resolved[i]<-NA
}
for(i in 1:length(Jira$Status)){
        if(Jira$Status[i] == "Resolved"){
          Jira$Resolved[i]<-Jira$Updated[i]
        }
}
```

```
Jira_Sprints <- Jira %>%
  select(
    Sprint,
    Sprint_1,
    Sprint_2,
    Sprint_3,
    Sprint_4
  )
```

```
for(i in 1:length(Jira$Status)){
  if(is.na(Jira$Sprint_4[i]) == FALSE){
  Jira$n_sprints[i]<-5
  }
  else if(is.na(Jira$Sprint_3[i]) == FALSE){
    Jira$n_sprints[i]<-4
  }
  else if(is.na(Jira$Sprint_2[i]) == FALSE){
    Jira$n_sprints[i]<-3
  }
  else if(is.na(Jira$Sprint_1[i]) == FALSE){
    Jira$n_sprints[i]<-2
  }
  else if(is.na(Jira$Sprint[i]) == FALSE){
    Jira$n_sprints[i]<-1
  }
  else{
    Jira$n_sprints[i] <-0
  }
}
```

```
## Warning: Unknown or uninitialised column: `n_sprints`.
```

This code block updates the "Resolved" column. The resolved column only contains a value if the row's value in the Status column is "Resolved". If the logical statement is TRUE then the Resolved column's value is the same as the Updated column.

```
Jira <- Jira %>%
    mutate(
      Priority = recode(
       Priority,
          `1 - Trivial` = "R1",
          `Trivial` = "R1",
          `2 - Minor` = "R1",
          `Minor` = "R1",
          `3 - Major` = "R2",
          `Major` = "R2",
          `4 - Critical` = "R3",
          `Critical` = "R3",
          `5 - Blocker` = "R4",
          `Blocker` = "R4",
      ), # Rename each value to R1, R2, R3, and R4, the military's preferred terms.
      completed = recode(
        Status,
        Resolved = 1,
        Done = 0,
        Closed = 0,
        `In Progress` = 0,
        `Ready For Review` = 0,
        `Ready For Test` = 0,
        Reopened = 0,
        .default = 0
      ), # set the value of each resolved issue to 1 for computational purposes
      Parent = `Parent id`,
      Issue = `Issue id`,
      Type = `Issue Type`,
      date_created = as.Date(strptime(Created, format = "%m/%d/%Y %H:%M")),
      date_updated = as.Date(strptime(Updated, format = "%m/%d/%Y %H:%M")),
      date_resolved = as.Date(strptime(Resolved, format = "%m/%d/%Y %H:%M")),
      # read columns as dates instead of characters
      time_spent = date_updated - date_created,
      # create the time_spent parameter, which is computed by finding
      # the difference between the date of creation and last day updated
      repair_time = date_resolved - date_created
      # create the repair time parameter, which is computed by finding
      # the difference between the date of creation and day resolved
    )
```

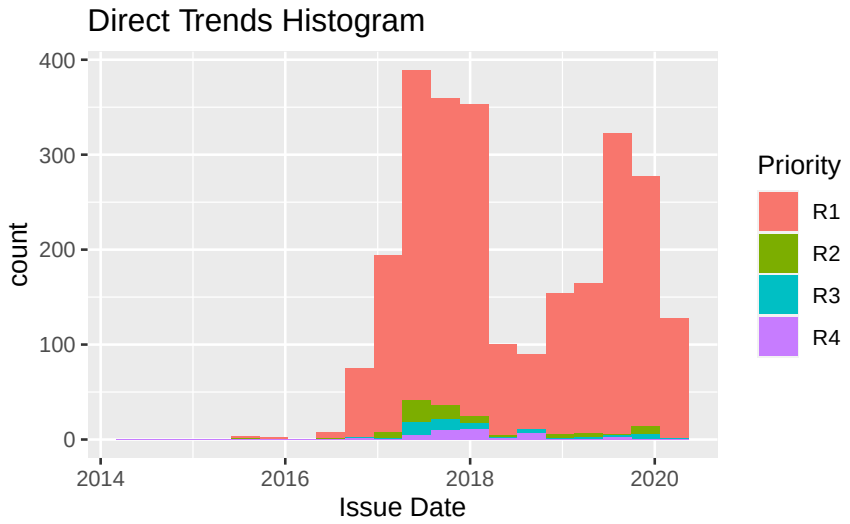
```
Jira_reduced <- Jira %>%
  select(
    Priority,
    completed,
    date_created,
    date_updated,
    date_resolved,
    time_spent,
    repair_time,
    Assignee,
    n_sprints
  )
```

This code block renames columns and makes them usable. At the end, it makes a new dataset with only the necessary columns.

```
Number_of_weeks <- (max(Jira_reduced$date_updated) - min(Jira_reduced$date_created))/7
Number_of_weeks <- as.numeric(Number_of_weeks)
```

This code block caluclates the number of weeks.

```
Jira_reduced %>%
  ggplot() +
  geom_histogram(aes(x = date_created, fill = Priority), bins = 20) +
  labs(title = "Direct Trends Histogram") +
  xlab("Issue Date")
```



This graph shows the Direct Trends as they are shown in the PowerPoint from the Navy. This graph shows the number of different issues reported, with the colors indicating the Priority of the issue.

```
Jira_reduced %>%
  group_by(Priority) %>%
  summarise(
    Defects = n(),
    `Defects in four weeks` = n() / (Number_of_weeks / 4),
    `MTTR` = (mean((repair_time), na.rm = TRUE)),
    DRE = sum(completed) / n()
  )
```

Priority	Defects	Defects in four weeks	MTTR	DRE
R1	2454	27.1911357	73.20888 days	0.4311328
R2	76	0.8421053	86.38710 days	0.4078947
R3	48	0.5318560	46.91304 days	0.4791667
R4	43	0.4764543	90.41667 days	0.8372093

This graph shows some summary statistics, most of which are estimated to a degree. Each parameter is calculated for each priority level.

- Defects- This column shows the total number of defects.
- Defects in four weeks- This column shows the average number of defects over four weeks.
- MTTR (mean time to repair)- This column shows the average number of days spent from the creation of the project to the day it was last updated (only in cases where the status of the project is "Resolved").
- DRE (direct removal efficiency)- This column shows the number of total issues resolved divided by the number of total issues.

```
Jira_reduced %>%
  group_by(n_sprints) %>%
  summarise(
    Defects = n(),
    `Defects in four weeks` = n() / (Number_of_weeks / 4),
    `MTTR` = (mean((repair_time), na.rm = TRUE)),
    DRE = sum(completed) / n()
  )
```

n_sprints	Defects	Defects in four weeks	MTTR	DRE
0	1095	12.1329640	177.00000 days	0.0009132
1	1244	13.7839335	68.19429 days	0.7033762
2	238	2.6371191	91.43478 days	0.9663866
3	41	0.4542936	87.74359 days	0.9512195
5	3	0.0332410	56.00000 days	1.0000000