Metrics and Their Usefulness in Team Software Development

Samuel Raydon LaGrave

Follow this and additional works at: https://louis.uah.edu/honors-capstones

Recommended Citation
https://louis.uah.edu/honors-capstones/455

This Thesis is brought to you for free and open access by the Honors College at LOUIS. It has been accepted for inclusion in Honors Capstone Projects and Theses by an authorized administrator of LOUIS.
Metrics and Their Usefulness in Team Software Development

by

Samuel Raydon LaGrave

An Honors Capstone

submitted in partial fulfillment of the requirements

for the Honors Diploma

to

The Honors College

of

The University of Alabama in Huntsville

April 27, 2020

Honors Capstone Director: Dr. Harry Delugach

Associate Professor of Computer Science

04/27/2020

Student Date

Director Date

Department Chair Date

Honors College Dean Date

Honors College

Frank Franz Hall

+1 (256) 824-6450 (voice)

+1 (256) 824-7339 (fax)

honors@uah.edu
Honors Thesis Copyright Permission

This form must be signed by the student and submitted as a bound part of the thesis.

In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

Samuel Raydon LaGrave

Student Name (printed)

[Signature]

Student Signature

04/27/2020

Date
# Table of Contents

**Abstract** 2

**Introduction** 3
- What are Metrics 3
- What Metrics were used in this Project 4

**Team Metrics** 5
- Meeting Details 5
- Word Count of Documentation 7
- Lines of Code 9
- Interpersonal Messages 11

**Individual Metrics** 13
- Hours Recorded 14
- Repository Commits Over Time 21

**Knowledge Gained** 22
Abstract

The importance and usefulness of metrics has long been a point of debate within the world of software development. Some prominent developers argue that the quantifiable data provides useful insight, while others suggest that the effort required to gather the data outweigh the supposed insight. This document provides an overview of my Software Development team's recorded metrics and our experiences and thoughts on their usefulness.

After first discussing the purpose of recording metrics, this document will present the data that was collected during my team's time working on our Team Software Development project. Following that are my thoughts on the process as a whole and if I feel metric tracking is a worthwhile task for students in a similar position to undertake.
Introduction

In the Spring 2020 semester I participated in a semester long team software development project for my CS 499 course. Throughout my discussions with the instructor of the course, Dr. Delugach, we lingered several times on the topic of metrics in software. We decided that for my Honors Capstone I would, with the help of my fellow team members, collect several team and individual metrics, evaluate those metrics, and write a paper discussing my findings. Before presenting the data, there are a few topics that must be discussed first.

What are Metrics

Merriam-webster defines metrics as "a standard of measurement". They are a statistic about a portion of software development that is measurable. They allow a developer or team of developers to analyse previous work, optimize current development, and better plan for future projects.

Simply because an aspect of software development is measurable does necessarily not make it a useful metric. A quote frequently used when discussing metrics, often attributed to Albert Einstein, is "Not everything that can be counted counts, and not everything that counts can be counted". The first portion of the statement tells us that because a metric exists does not inherently mean it is valuable. The other part explains that not all important aspects of software development can be quantified. Because of these concepts, a large portion of properly utilizing metric tracking is determining the proper metrics for your situation.

The usefulness of metric tracking is not something that everyone agrees on. While many people in the field feel that metrics provide useful insight into the development process, others
feel that the cost of time spent recording and analysing the metric data outweighs any supposed gain in insight. After presenting the data collected throughout my project, I will discuss my thoughts on the topic based on my experiences with this project.

What Metrics were used in this Project

The metrics collected during this project were split into two categories: team metrics and individual metrics. Team metrics track the progression of the project as a whole along with details of interpersonal interactions. Individual metrics provide insight into each developer's personal progress and workflow.

Team metrics collected include:

- Number of topics per meeting
- Meeting date and time
- Meeting attendance
- Word count of documentation
- Lines of code
- Digital messages sent between members

Individual metrics collected include:

- Time spent working on the project
- Commits over time
Team Metrics

This section of the report contains the team metrics data collected throughout the project, along with understanding that can be gained from the data. It also contains details of how the team collected said data, and how we made use of the information gained from it.

Meeting Details

As mentioned, details such as attendance, number of items discussed, and date/time were collected from each of the team's meetings.

Figure 1 visualizes the number of items discussed at each meeting. The team originally believed that this value would be higher towards the beginning of the project and decrease as the project went on. We believed this because the beginning of the semester was filled with meetings covering a wide range of topics due to us planning the entire project. We believed meetings would become more focussed as we began developing systems, leading to meetings with less topics. Obviously that is not what happened - throughout the semester we did not see the expected decline in number of topics. It should be noted that the sporadic nature of the graph
may be due somewhat to the fact that the job of recording topics rotated between team members throughout the semester. If this project were to be repeated, I feel like having one person constantly recording this metric each meeting would lead to more consistency.

Figure 2 shows how long each meeting was. Most of the meetings hovered around the 15-30 minute time frame, which is where the team tried to keep them; we felt like having shorter meetings helped keep us on topic. There are obviously several notable exceptions to that rule. Those were meetings that handled a specific task and often included work being done during the meeting, which was not standard procedure for our meetings. An example would be the Jan 20th meeting, which was spent preparing our team's software design plan and corresponding presentation. Another example would be Apr 16th, where the team merged all the outstanding branches and produced the first packaged product.
Figure 3 shows how many team members were in attendance at each meeting. For almost all meetings, all team members were present. It is important to mention that one of the team members dropped the course (and left the team) later in the semester, which can be seen in the shift from four members consistently to three around the 25th of March. Those three remaining outliers were due to scheduling conflicts, or in the case of the Mar 18th meeting, only two members were needed to discuss the topics proposed. This figure shows that our team did well at choosing meeting times that worked for all of the members, which was one of our goals early on.

**Word Count of Documentation**

The total word count of documentation was also recorded as a team metric. This value includes documents such as the team's software design plan, backlog, the sprint review reports, and other documentation.
Words of Documentation Written

Fig. 4

Words of Documentation Written

Fig. 5
Figures 4 and 5 show the same data in slightly different representations; they both show how many words of design documentation had been written by specific dates. When compiling this data I had a hard time selecting a graph type for this data. This is because this data was not something that was continuously collected - an updated word count was recorded after each document was submitted. This caused the data to look very much like the stair graph shown in figure 4. However, that representation makes it look like the words of documentation were all written at one point in time, which isn't accurate. Most documents were written over several iterations, covering approximately 3-4 days per document. The data representation in figure 5 shows this steady growth better, but it shows a gradual increase in size between each submission, which is also not 100% accurate. A better representation of this data would be a mixture between the two figures, showing a mostly plateau graph, with a gradual increase a few days before each marked submission date. However, with the data representation tools I have at my disposal, such a graph is not possible to produce.

The data received from this metric shows that our team continually wrote documentation throughout the semester. Of particular note is the large increase in documentation recorded around April 21st and onward. This was the creation of the team's final deliverable document, along with the team's final sprint report. The final deliverable included most of the content from several of the previous deliverable documents, along with new content from the team's testing phase.

**Lines of Code**

While the merits of metric recording may itself be questioned, this metric stands as most likely the most controversial in terms of insight given. Many people argue that comparing
developers' 'lines of code written' statistics is not beneficial, as that value is strongly influenced by coding style, format, and practices. Because of this, I chose lines of code to be used as a team metric, allowing us to see the growth of the project over time, taking into account style and formatting habits from the entire team.

![Graph showing total lines of code](image)

**Fig. 6**

Figure 6 shows the lines of code calculated from all of the commits made to the repository. There are a few periods of interest in the graph, the first of which is the portion with two large jumps and no change between (approx. Jan 13-Feb 9). The first jump was the creation of the repository and using some tutorial to familiarize ourselves with the library we were using. The large plateau was the period of time the team was focussed on writing the design document and planning the semester. The second jump in lines of code represents the creation of the base classes and changing the starter code to align with our needs.
The next period of interest is approximately the month of March. That period of time was marked by a lot of code deletion due to the team rewriting and redeveloping a lot of the code behavior. This is demonstrated well on the graph by the fact that several times the total code length goes down, which happens during no other time frame. The last period is the month of April, marked by consistent growth. This represents the team pushing to finalize the product as the due date was late April.

**Interpersonal Messages**

The final metric used to measure the team's progress was the number of messages sent between members. To fully understand this metric, a few facts of note should be mentioned:

1. Our team used Discord to communicate remotely
2. Team member names are anonymized - this is discussed more in the individual metrics section
3. This project took place in the Spring 2020 semester, which was notably disturbed by the COVID-19 pandemic. Because of this, this metric was split into two portions to see how the team adapted to working purely remotely after the university was closed.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Multiple</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Member A</strong></td>
<td>-</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>29</td>
<td>37</td>
</tr>
<tr>
<td><strong>Member B</strong></td>
<td>6</td>
<td>-</td>
<td>16</td>
<td>12</td>
<td>27</td>
<td>61</td>
</tr>
<tr>
<td><strong>Member C</strong></td>
<td>0</td>
<td>16</td>
<td>-</td>
<td>3</td>
<td>53</td>
<td>72</td>
</tr>
<tr>
<td>Member A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>Multiple</td>
<td>Total</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>----------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>46</td>
<td>0</td>
<td>7</td>
<td>16</td>
<td>69</td>
<td></td>
</tr>
<tr>
<td>Member B</td>
<td>43</td>
<td>-</td>
<td>2</td>
<td>12</td>
<td>91</td>
<td></td>
</tr>
<tr>
<td>Member C</td>
<td>0</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Member D</td>
<td>15</td>
<td>13</td>
<td>0</td>
<td>-</td>
<td>59</td>
<td></td>
</tr>
</tbody>
</table>

Table 1 - Team communication before online-only

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Multiple</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member A</td>
<td>-</td>
<td>46</td>
<td>0</td>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>Member B</td>
<td>43</td>
<td>-</td>
<td>2</td>
<td>12</td>
<td>34</td>
</tr>
<tr>
<td>Member C</td>
<td>0</td>
<td>2</td>
<td>-</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Member D</td>
<td>15</td>
<td>13</td>
<td>0</td>
<td>-</td>
<td>31</td>
</tr>
</tbody>
</table>

Table 2 - Team communication after online-only

The preceding two tables present the number of messages each team member sent to each other member. Each row represents the sender with the columns representing the primary recipient. If a message was meant for multiple members or the entire time, it was recorded in the "Multiple" column.

The information given by this metric provides some interesting insight on how our team used digital communication. Before going online-only, our team sent 251 messages between the members, compared to 223 after going online-only. These numbers alone may make it seem like we communicated less after the switch, which is not the case. There are a few factors that cause the data to look that way. The first of which is the fact that one of the members stopped communicating with the team due to them dropping the class early into the "after" period.
Another thing is the length of time. The Discord server was created on January 13th and the transition is considered to have happened on March 13th, giving 61 days in the "before" period, compared to 40 days in the "after" period (April 22nd was the last recorded message). When taking this into account each team member sent approximately 1.02 messages per day before going online only and 1.83 messages per day after going online-only.

This increase in short form communication correlates with a decrease in long form communication (meetings) which can be seen in figures 1-3 where meetings are much more frequent before the transition on the 13th of March.

**Individual Metrics**

The metrics used to track individual progress throughout the progress were hours spent working on the project and commits over time. Because Dr. Delugach was the instructor for the class in addition to being this project's director, we worried my teammates may be inclined to embellish their time metrics if they believe their grade was swayed by that information. To dissuade this, it was decided early on that all metrics involving individual members would be anonymized. Dr. Delugach had also stated several times these metrics would in no way affect our grades for CS 499.
Figure 7 shows the amount of time each team member contributed to the project. Despite frequent reminders throughout the semester, some members consistently slacked on recording their hours, which results in the rather lopsided graph of figure 7.
Member A's time categorization

- Programming: 82.4%
- Meeting: 13.3%
- Documentation: 4.3%

38:15

Member A's time by weekday

- Wednesday: 45.3%
- Tuesday: 3.2%
- Monday: 14.6%
- Friday: 7.5%
- Thursday: 7.3%
- Saturday: 12.4%
- Sunday: 9.7%

21:00

Fig. 8

Fig. 9
Team member A’s recorded time is further divided in figures 8 & 9. Figure 8 shows how their 46 hours and 24 minutes were divided between the categories Meeting, Documentation, and Programming. Figure 9 displays how the same 46:24 was spread over the weekdays. Member A had a very clear preference for working on Wednesday - this was most likely influenced by the fact that Wednesday was a class day and we often had meetings on Wednesdays as well. Following that, Monday has the next most time, which like Wednesday was a class day and frequent meeting day for our team.

![Member B's time categorization](image)

Fig. 10
Once again, team member B's time is categorized in two figures. Figure 10 shows their time spent categorized by the type of work they were doing. Note this graph shows that member B made use of the research category while recording their time data. Figure 11 shows that member B's time was more spread out over the week compared to A.
Again, figures 12 and 13 breakdown team member C's time. Once again, Monday and Wednesday stand out as days with larger than average amounts of time spent. This is expanded upon since meetings make up a larger percentage of member C's time than the previous two members.

**Member D's time categorization**

- Programming: 28.5% (3:32)
- Documentation: 32.8% (3:00)
- Meeting: 38.7% (2:36)

Fig. 14
Finally we have the categorization of member D. Similar to the majority of the team, member D spent a lot of time working on this project on Monday compared to other weekdays. Figure 14 is the most evenly split time categorization graph, showing that member D's time was pretty evenly split between writing code, documentation, and attending meetings.

All of these graphs show that most of the work happened on either Monday or Wednesday. As previously mentioned, these were the days we had classes, so all of the members had the time reserved for this project already. It is also interesting to note the trends among the other days. For example, member A spent a fair amount of time working on the project on Saturdays, whereas the other three have very little to no time recorded on Saturdays.
Repository Commits Over Time

**Member A's commits over time**

![Graph showing Member A's commits over time from 1/26/2020 to 4/5/2020.]

**Fig. 16**

**Member B's commits over time**

![Graph showing Member B's commits over time from 1/26/2020 to 4/5/2020.]

**Fig. 17**

**Member D's commits over time**

![Graph showing Member D's commits from 2/15/2020 to 3/29/2020.]

**Fig. 18**
Figures 16-18 visualize the number of commits to the repository that each member made. Team member C does not have an associated graph due to the fact that they did not make any commits to the project's repository.

The difference in committing style is clearly displayed in these three figures. Team member B would write/edit code until a usable portion of a feature was achieved and then commit that. Team member A would commit their work only when a feature was completely working, making them have less total commits than member B. Team member D would work on several systems, often over a few days, and then make one massive commit. While member A and B's committing habits may have been different, they had no problem collaborating. While member D's habit would not have been the best for collaborative coding, they often worked on stand alone systems that were unrelated to other systems being worked on, meaning their style did not cause many problems.

**Knowledge Gained**

The primary goal of this project was to see if the knowledge gained from recording and analysing these metrics provided useful insight into the teams process. After reviewing this data, the following guidelines might have beneficial for the team:

- Team meetings should be kept around 15-30 minutes for a standard meeting, and around 1:30 for a longer meeting in preparation for a presentation or deliverable.
- There should be between 5 and 10 items discussed per meeting. Any less could likely have been handled via messages. Any more may cause issues to be rushed.
• If possible, meetings should be held on Monday or Wednesday, and should be avoided on Saturday and Sunday.

• Team members should strive to work on the project consistently throughout the week. This helps avoid a lot of new code entering the project at once, increasing the chance of bugs arising.

• Every team member should check the Discord at least twice a day, minimizing the chance of a message going unnoticed for a long time.

• Team members should not write code for longer than an hour without committing. This will help avoid having multiple members edit the same chunk of code at once, leading to fewer merge issues.

I feel that recording these metrics and analysing them did give me a deeper understanding of what our team did right, and what we could have improved on. These metrics were all compiled and charted after the project was finished, allowing me to have all of the data at once. If I were to make use of metrics in future projects, I would compile and analyse the data more frequently. Since it does take a fair amount of time to evaluate the data, this should not be something undertaken daily, or even weekly. This would be a biweekly or monthly task, likely aligning with the end of a sprint.
Dear colleagues,

I apologize for the last-minute report, but you can well imagine how chaotic this particular end-of-semester is turning out.

Samuel's project was excellent! He undertook to monitor a small (4 person) software development team in CS 499 while gathering measurements on communications and work produced by the team members. In spite of obstacles, he produced a fine report. Not only did he display a good understanding of how metrics work in software development processes, he also produced some interesting results. As "luck" would have it, he was able to compare metrics before and after going to the all-online format. Not surprisingly, the team exchanged almost exactly twice the number of messages after the switch. It is valuable to quantify these effects.

He was also faced with what to do when a team member dropped out of the course and his team. His ability to adapt is commendable.

Please let me know what paperwork you will need from me on this.

Thanks,

Harry

Harry S. Delugach, Ph.D.

Associate Professor
Computer Science Dept.
OKT N-351
University Of Alabama In Huntsville
Huntsville AL 35899 U.S.A.
voice: +1 256.824.6614 fax: +1 256.824.6239
delugach@uah.edu
http://www.cs.uah.edu/~delugach