Software Engineering

MEASURING SOFTWARE ENGINEERING

Suman Neupane

17309057

Table of Contents

Measuring Software Engineering

Introduction

One of the big underlying questions in the field of software engineering is identifying the difference between a good and a bad software engineering practice. In this report I will discuss the different ways of measuring the value of software engineering. Before outlining the ways an engineer can be measured we need to know what software engineering is. Software engineering is defined as "the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software" by IEEE Standard Glossary of Software Engineering Terminology.

Why do we measure software engineers? What does it mean to measure software engineering? How can it be done practically? What are the technologies that are used to measure the data? What are the techniques to process the data as well as their strengths and weaknesses? How can the technologies be used safely and reliably? Is it ethical? These are the questions raised when we talk about measuring a software engineer. I will attempt to give my opinion on each of the question above in this report.

What are the measurable data?

These are several ways in which the performance of a software engineer can be measured. Below are some of the metrices that are measurable.

Lines of code (LOC)

One of the oldest and the most common ways to measure the work done by a software engineer is by observing the total lines of code they write. Bill Gates once said, "Measuring programming progress by lines of code is like measuring aircraft building process by weight[1]". This might sound like a fair measurement, but it has many more observable disadvantages than advantages. Some of the advantages are automation of counting the lines and an intuitive metric where LOC helps measure the size of the software. Unified Code Count (UCC) is a software developed by USC and CSSE is an automated software lines of code counter which also provides Cyclomatic Complicity (Hira, 2015).

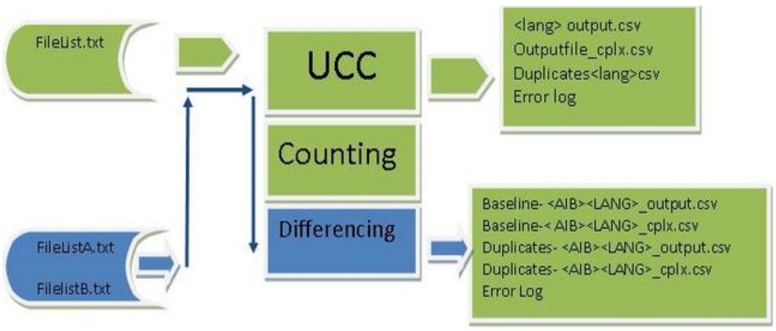


Figure 1: https://commons.wikimedia.org/wiki/File:Block_Diagram_of_UCC.JPG

Lack of accountability, lack of cohesion, adverse impact on estimation, etc. are some of the disadvantages of LOC. In my opinion difference in language is the biggest disadvantage and it is also biased towards languages such as Python. For example, John is an engineer who programs in python and Mark is an engineer who programs in C#. If both John and Mark decided to write simple hello world program the following would be the result.

Hello world in Python:

```
print ("Hello World")
```

Hello world in C#:

```
using System;
class Program
```

[1] "Measuring programming progress by lines of code..." - Programming Quotes (stormconsultancy.co.uk)

```
{
  static void Main (string [] args)
  {
    Console.WriteLine("Hello, world!");
  }
}
```

John’s program is 1 line and it prints the hello world where as Mark’s program is 8 lines and also prints hello world. This doesn’t mean Mark’s program is better than John’s program because both produce the same result. Therefore, in my opinion measuring an engineer by lines of code is not the best way to measure software engineers.

Code Coverage or Testing

Another technique of measuring software engineering is the testing and the code coverage. (Pittet, 2020). The following are the most common metrics that usually mentioned in the coverage report:

Functional coverage

- Total numbers of functions defined that have been called.

Statement coverage

- Total number of the statements in the program that have been executed.

Branches coverage

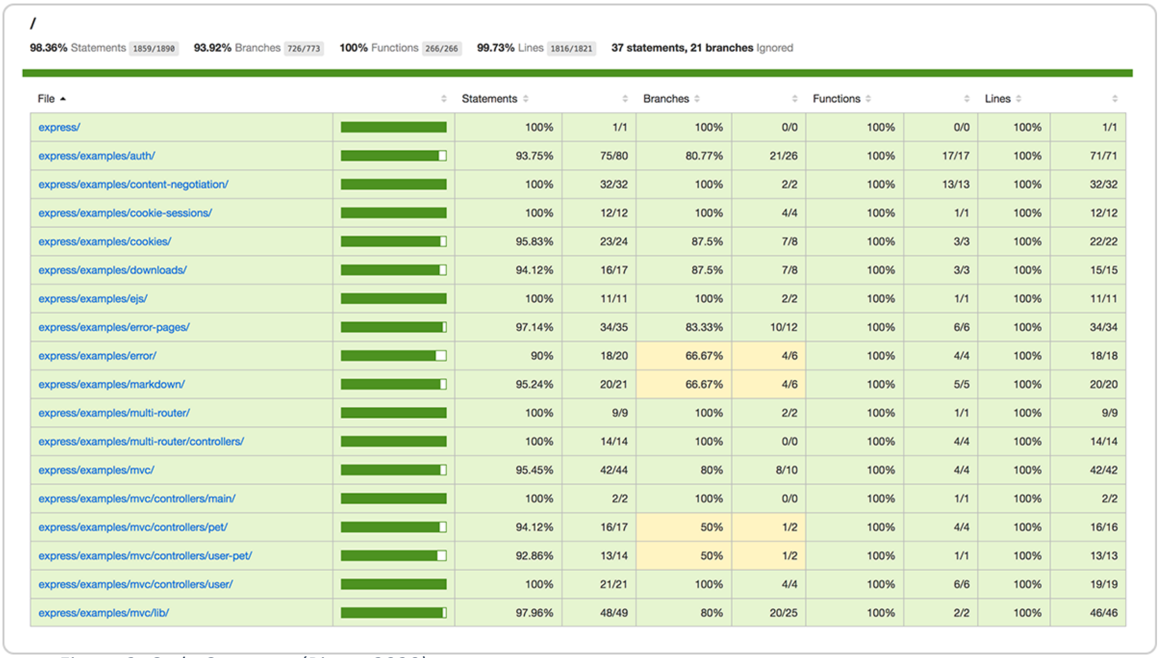
- Total numbers of branches of the control structures that have been executed.

Condition coverage

- Total number of Boolean sub-expressions that have been tested for true or false value.

Line coverage

- Total numbers of lines that have been tested.



/

98.36% Statements 1859/1890 **93.92%** Branches 726/773 **100%** Functions 266/266 **99.73%** Lines 1816/1821 **37 statements, 21 branches** Ignored

File ▲		Statements		Branches		Functions		Lines	
express/		100%	1/1	100%	0/0	100%	0/0	100%	1/1
express/examples/auth/		93.75%	75/80	80.77%	21/26	100%	17/17	100%	71/71
express/examples/content-negotiation/		100%	32/32	100%	2/2	100%	13/13	100%	32/32
express/examples/cookie-sessions/		100%	12/12	100%	4/4	100%	1/1	100%	12/12
express/examples/cookies/		95.83%	23/24	87.5%	7/8	100%	3/3	100%	22/22
express/examples/downloads/		94.12%	16/17	87.5%	7/8	100%	3/3	100%	15/15
express/examples/ejs/		100%	11/11	100%	2/2	100%	1/1	100%	11/11
express/examples/error-pages/		97.14%	34/35	83.33%	10/12	100%	6/6	100%	34/34
express/examples/error/		90%	18/20	66.67%	4/6	100%	4/4	100%	18/18
express/examples/markdown/		95.24%	20/21	66.67%	4/6	100%	5/5	100%	20/20
express/examples/multi-router/		100%	9/9	100%	2/2	100%	1/1	100%	9/9
express/examples/multi-router/controllers/		100%	14/14	100%	0/0	100%	4/4	100%	14/14
express/examples/mvc/		95.45%	42/44	80%	8/10	100%	4/4	100%	42/42
express/examples/mvc/controllers/main/		100%	2/2	100%	0/0	100%	1/1	100%	2/2
express/examples/mvc/controllers/pet/		94.12%	16/17	50%	1/2	100%	4/4	100%	16/16
express/examples/mvc/controllers/user-pet/		92.86%	13/14	50%	1/2	100%	1/1	100%	13/13
express/examples/mvc/controllers/user/		100%	21/21	100%	4/4	100%	6/6	100%	19/19
express/examples/mvc/lib/		97.96%	48/49	80%	20/25	100%	2/2	100%	46/46

Figure 2: Code Coverage (Pittet, 2020)

Number of commits

Another way to measure a software engineer is by processing the number of commits they have over a certain time frame. For example, daily, weekly or monthly time frames. In my opinion this can also be misleading and biased as LOC, but if the individual has a consistent number of commits then it can be a strong assessment of work done by the individual.

Velocity

Velocity is another method of measuring software engineering which is mostly used in agile software engineering. There are many different methodologies used to measure the productivity and efficiency of a team. Scrum methodology is one which includes the built-in way of measuring software development efficiency and overall productivity at a team level using the team's velocity. It also helps the team to estimate the amount of work required for completion in a given time frame based on their previous team projects.

Closed Tickets

Another technique of measuring software is the ticketing system which logs the work done by each individual in a group or a project (Osbourn, 2019). Counting the number of tickets that the engineer has closed can help the company accurately see what work is getting closed off. In my opinion, measuring the individuals work by the number of tickets they close is one of the best ways to measure software engineers. Since GitHub, Jira or Trello are some of the systems that engineers use to work on the tickets, it makes things easier for the company or manager to view the completed/closed tickets easily.

Why do we measure Software Engineers?

The procedure of collecting measures should be adapted to each organization and the subsequent implementation may take different forms. Depending on the nature of the measures, the researcher’s knowledge about what is to be measured, the purpose of collecting and on the appropriateness of the instruments (Bellini, 2008). There are several ways an origination can measure their employee’s work. Efficiency & productivity of the engineer is one of the main reasons behind the measurement of the individual software engineer.

Efficiency & Productivity of Engineer

Efficiency is one of the fundamental reasons to measure software engineers because it helps the company to assess employee performance. Efficiency is measured by the percentage of an individual’s contributed code that is productive. Higher efficiency rates mean that the code will be beneficial for the company in the long run. Software productivity metrics terminology is given to ensure an understanding of measuring the data for both the code and the document (IEEE, 2002).

Measuring software engineers is not an easy task since there are many methodologies which engineers use these days. An example of this being agile programming. In agile programming engineers must quickly adapt to the changes made by clients, as it is their highest priority to satisfy the costumer and to deliver the valuable software continuously (Eby, 2017). This sometimes results in individuals taking longer to produce the code which could delay the project.

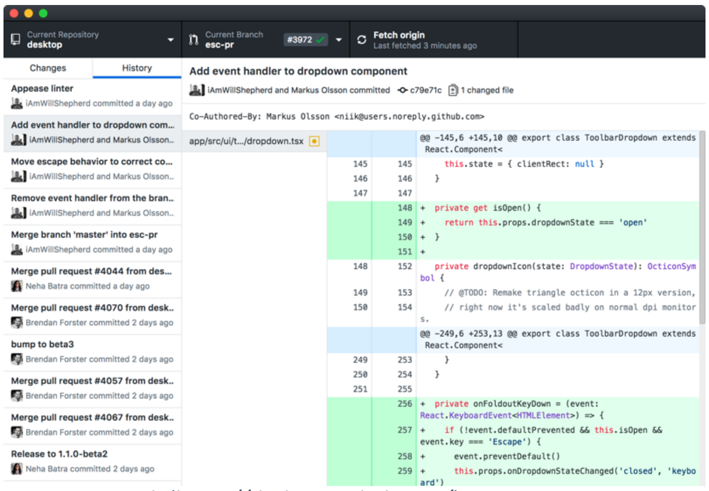
Therefore, measuring software engineering is vital to recognise the work done by the individual and to conclude their performance in the company.

Technologies

It is not easy to evaluate the effectiveness of technologies used on an actual software development. There are many factors which could influence the programmer, effectiveness and problem complexity are few of the examples (D.N. Card, 1987).

GitHub

GitHub is probably the most used platform to assess the progress of the software engineers. It is also widely used as a version control, to store the code as well as to track and control the changes. Version control helps engineers to track their progress and manage the changes that they make to the code. It not just helps the engineers it also helps the company or manager to assess the progress made by each individual. This makes it easy to view and compare who has done the most work in the project. This can be done by looking at their commits, number of closed tickets and the velocity of the code.



Appease linter
iAmWillShepherd committed a day ago
Add event handler to dropdown com...
iAmWillShepherd and Markus Olsson...
Move escape behavior to correct co...
iAmWillShepherd and Markus Olsson...
Remove event handler from the bran..
iAmWillShepherd and Markus Olsson...
Merge branch 'master' into esc-pr
iAmWillShepherd committed a day ago
Merge pull request #4044 from des...
Neha Batra committed a day ago
Merge pull request #4070 from desk..
Brendan Forster committed 2 days ago
bump to beta3
Brendan Forster committed 2 days ago
Merge pull request #4057 from desk..
Brendan Forster committed 2 days ago
Merge pull request #4067 from desk..
Brendan Forster committed 2 days ago
Release to 1.1.0-beta2
Neha Batra committed 2 days ago

Add event handler to dropdown component
iAmWillShepherd and Markus Olsson committed c79e71c 1 changed file

Co-Authored-By: Markus Olsson <niik@users.noreply.github.com>

app/src/ui/t.../dropdown.tsx

```
@@ -145,6 +145,10 @@ export class ToolbarDropdown extends
 React.Component<
      this.state = { clientRect: null }
    }

+  private get isOpen() {
+    return this.props.dropdownState === 'open'
+  }
+
   private dropdownIcon(state: DropdownState): OcticonSym
bol {
     // @TODO: Remake triangle octicon in a 12px version,
     // right now it's scaled badly on normal dpi monitor
s.
@@ -249,6 +253,13 @@ export class ToolbarDropdown extends
 React.Component<
      }
    }

+  private onFoldoutKeyDown = (event:
React.KeyboardEvent<HTMLElement>) => {
+    if (!event.defaultPrevented && this.isOpen &&
event.key === 'Escape') {
+      event.preventDefault()
+      this.props.onDropdownStateChanged('closed', 'keybo
ard')
```

Figure 3: GitHub (https://desktop.github.com/)

TimeFlip

TimeFlip is a small physical gadget that lets the user easily track how much time they spend on different activities just by flipping it around (TimeFlip). It is dodecahedron in shape which has a tiny accelerometer inside that can tell the user which way it is sitting. It is very simple to operate, all we have to do is flip it around so the side corresponding to what we are doing is facing up. The chip inside then sends a signal to the app on our phones over Bluetooth. This technology is not just for measuring

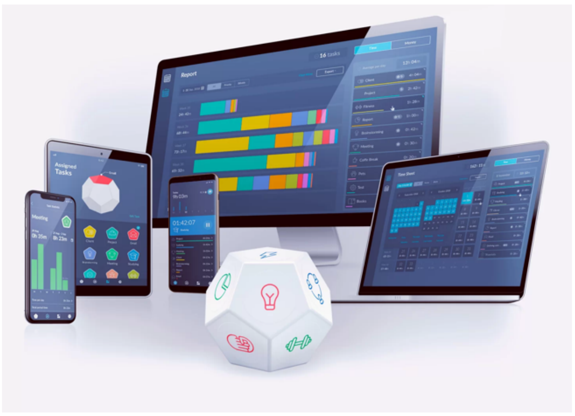
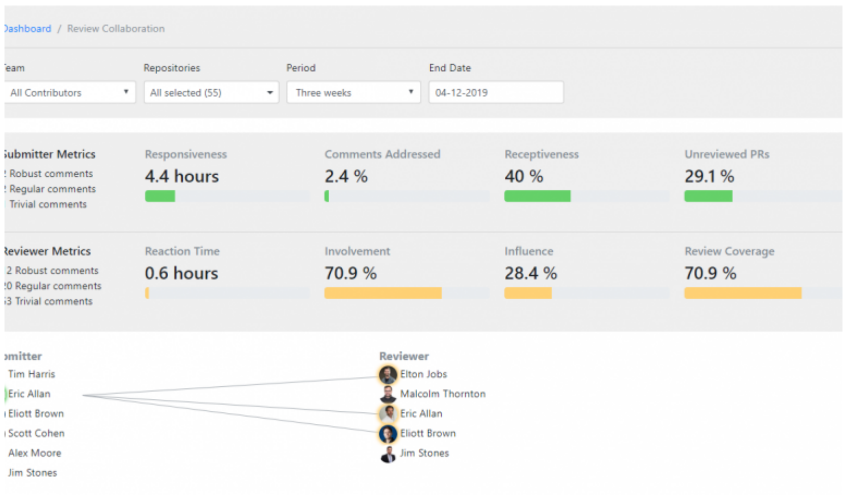


Figure 4: TimeFlip (https://timeflip.io/)

software engineering, it can be used by anybody to efficiently track and manage their time.

Waydev

Waydev is a fairly new agile data-driven software that helps engineers track their output directly in the git repository without their manual input. It analyses their codebase from Git platforms like GitHub, Gitlab, Azure DevOps & Bitbucket to help bring out the best in their software engineering projects. Waydev helps the company or manager to gain useful insights about the works and activities that are being performed by software engineering groups or particular individuals. It is said to be very accurate when it comes to generating the reports as it has the functionality to auto generate reports without any inputs. It gives engineers a productivity insight, for example, calculating the ratio between time spent by the engineer working on a certain project and the total number of commits made between that time period.



ashboard / Review Collaboration

eam	Repositories	Period	End Date
All Contributors	All selected (55)	Three weeks	04-12-2019

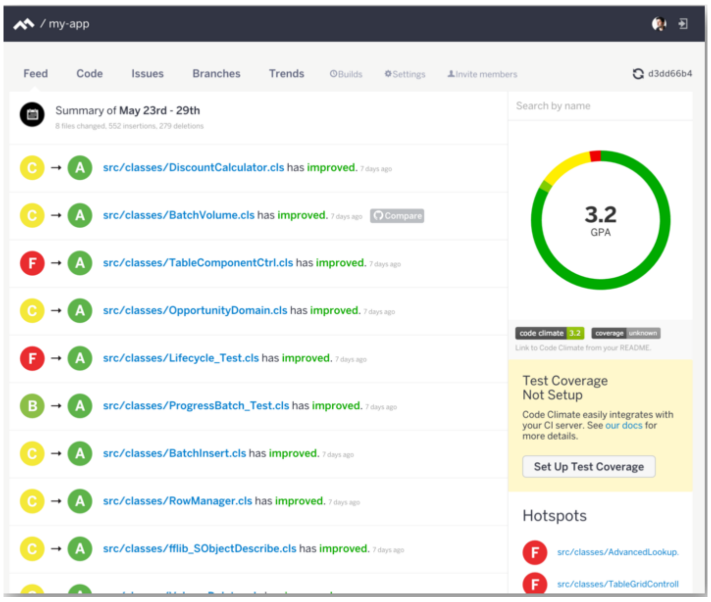
ubmitter Metrics	Responsiveness	Comments Addressed	Receptiveness	Unreviewed PRs
Robust comments Regular comments Trivial comments	4.4 hours	2.4 %	40 %	29.1 %

eviewer Metrics	Reaction Time	Involvement	Influence	Review Coverage
2 Robust comments 0 Regular comments 3 Trivial comments	0.6 hours	70.9 %	28.4 %	70.9 %

Figure 5: Waydev (https://waydev.co/review-collaborate/)

Code Climate

Another tool software engineers use to take control over their code quality is Code Climate. It helps engineers to configure the test coverage as well as the data maintainability throughout the workflow development (CodeClimate). As we know, our codes can have some subtle issues which wouldn't be obvious at first glance, which is why the Code Climate helps engineers to trap these issues by analysing each pull request before it is integrated.



Summary of May 23rd - 29th
8 files changed, 552 insertions, 279 deletions

C → A src/classes/DiscountCalculator.cls has improved. 7 days ago
C → A src/classes/BatchVolume.cls has improved. 7 days ago Compare
F → A src/classes/TableComponentCtrl.cls has improved. 7 days ago
C → A src/classes/OpportunityDomain.cls has improved. 7 days ago
F → A src/classes/Lifecycle_Test.cls has improved. 7 days ago
B → A src/classes/ProgressBatch_Test.cls has improved. 7 days ago
C → A src/classes/BatchInsert.cls has improved. 7 days ago
C → A src/classes/RowManager.cls has improved. 7 days ago
C → A src/classes/fflib_SObjectDescribe.cls has improved. 7 days ago

Link to Code Climate from your README.

Test Coverage Not Setup
Code Climate easily integrates with your CI server. See our docs for more details.
Set Up Test Coverage

F src/classes/AdvancedLookup.
F src/classes/TableGridControll

Figure 6: Code Climate (https://codeclimate.com/changelog/57504dc875b32312a30045e8/)

By using software like GitHub, engineers can know straightway when the quality of a code has been changed, encouraging them to click and see the full analysis of results on the software. Code Climate has some incredible features such as Test Coverage, Integration, Status and Tracking.

Algorithmic Approach

Halstead Metrics

Maurice Howard Halstead introduced the measurement for measuring the complexity of the program in 1977 (T. Hariprasad, 2017). We can't assume that a program with minimum lines of code has a less multifaceted nature than a program with extra lines of code. Sometimes, even a big program can be basic and the smaller program can be more mind boggling. Halstead's metric measurements mostly depend on program execution and the measures, which are then analysed specifically from the operators and operands from source code. Halstead uses the following measurements (T. Hariprasad, 2017):

- n1 are number of unique or distinct operators in the code.
- n2 are number of unique or distinct operands in the code.
- N1 are the total number of occurrences of operators in the code.
- N2 are the total number of occurrences of operands in the code.

The following equation measure the length of the program:

$$N = N1 + N2$$

The following equation measure the vocabulary of the program:

$$n = n1 + n2$$

The following equation measure the volume of the program:

$$V = (N1 + N2) log2(n1 + n2) = N\ log2(n)$$

The following equation measure the difficulty of the program:

$$D = \left(\frac{2}{n1}\right) * \left(\frac{n2}{N2}\right)$$

Constructive Cost Model (COSYSMO)

COSYSMO was introduced in 2005 to estimate the cost of the project. This algorithm helps the company to estimate the cost of their project, if the estimated cost is too big the company could end up wasting capital (Alstad, 2019).

- Top-Level Estimating Equation

$$PH = A.(AdjSize)^E . \prod_{j=1}^{15} EM_j$$

- Size Model

$$AdjSize x_{c3} = \sum_{SizeDrivers} \begin{matrix} eReq\big(Type(SD), Difficulty(SD)\big) \times \\ PartialDevFactor\big(AL_{Start}(SD), AL_{End}(SD), RType(SD)\big) \end{matrix}$$

- Exponent Model

$$E = E_{Base} + SF_{ROR} + SF_{PC} + SF_{RV}$$

Ethical Concern

I personally think gathering engineer's data to assess their performance can be ethical since most of the measuring is done based on the code provided by the engineer. Nowadays, companies have to be extra careful with the data that they gather since the introduction of GDPR Guidance in 2016. It is a regulation introduced to regulate the data protection and the privacy of all the members of the European Union and the European Economic Area.

Any data collected by the companies have to be stored safely since it may contain sensitive information about their employees. The IEEE Computer Society and the ACM, two leading international computing societies, have introduced the Software Engineering Code of Ethics which was adopted in 2000. The code of ethics is intended as a guide for members of the evolving software engineering profession. The main principles of the code are as follows (IEEE, 2002): Public, Client and Employer, Product, Judgement, Management, Profession, Colleagues and Self. Engineers, Companies and Clients all have to keep these principles in mind while working on a project. Engineers also have to be careful on the type of data their software collects from its users.

Tools like GitHub, Code Climate, Waydev etc only gathers the data that the engineer has consented to. There are many other ways in which companies gather data from their employees, some examples include Steelcase chair, Humanyze, Status Today and CCTV cameras. I personally find these kinds of monitoring systems unethical due to potential capturing of very sensitive data about their employees. This could have a negative effect on the productivity of their employees since every move they make is being recorded and in a sense 'judged'.

References

Alstad, J. P. (2019). Development of COSYSMO 3.0: An Extended, Unified Cost Estimating Model for Systems Engineering. In *Procedia Computer Science* (pp. 55-62).

Altavater, A. (2017, April 18th). *Measuring Software Development Productivity*. Retrieved from stackify.com: stackify.cim/measuring-software-developemnt-productivity/#:~:text=The%20primary%20purpose%20of%20measuring,in%20a%20high-efficiency%20rate.

Bellini, C. &. (2008). Measurement in software engineering: From the roadmap to the crossroads. *International Journal of Software Engineering and Knowledge Engineering*, 57.

CodeClimate. (n.d.). Retrieved from https://www.blissfully.com/code-climate/

Coldewey, D. (2018, January 30). *TimeFlip is a time-tracking gadget simple enough that I might actually use it*. Retrieved from https://techcrunch.com/: https://techcrunch.com/2018/01/30/timeflip-is-a-time-tracking-gadget-simple-enough-that-i-might-actually-use-it/?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuYmluZy5jb20v&guce_referrer_sig=AQAAAKLLROYC5E9pX84FHm289jqfmoTmWuZn421ijjvKpOMeFz_LOb89rdYWVHF47_i0

D.N. Card, F. M. (1987). Evaluating Software Engineering Technologies. In *IEEE Transactions on Software Engineering* (pp. 845-851).

Eby, K. (2017, Feb 15). *What's the Difference? Agile vs Scrum vs Waterfall vs Kanban*. Retrieved from https://www.smartsheet.com: https://www.smartsheet.com/agile-vs-scrum-vs-waterfall-vs-kanban

Hira, A. (2015, 11 08). */ All Podcast Series / CSIAC Webinars / Using Unified Code Count (UCC) for Data Collection*. Retrieved from https://www.csiac.org/: https://www.csiac.org/podcast/using-unified-code-count-ucc-for-data-collection/

IEEE. (2002). IEEE Standard for Software Productivity Metrics. *IEEE Xplore*.

Osbourn, T. (2019, 12 10). *Creative Ideas to Measure Developer Productivity*. Retrieved from https://textexpander.com: https://textexpander.com/blog/measure-developer-productivity/

Pittet, S. (2020, 11 20). *An introduction to code coverage*. Retrieved from https://www.atlassian.com/: https://www.atlassian.com/continuous-delivery/software-testing/code-coverage

T. Hariprasad, G. K. (2017). Software complexity analysis using halstead metrics. In *2017 International Conference on Trends in Electronics and Informatics (ICEI)* (pp. 1109-1113).

TimeFlip. (n.d.). Retrieved from https://timeflip.io/: https://timeflip.io/