Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

CSU33012
Measuring Software Engineering

Luke McGrath
17337376
23rd November 2020

1 – Measuring Software Engineering

1.1 – Introduction

In this essay I will discuss the topic of measuring software engineering. The discipline of software engineering can be measured in many ways, and I will discuss these methods and highlighting both the benefits and drawbacks associated with them. I will also discuss some different platforms that are availaboe to measure software engineering and the different algorithms that can be used to do so. Finally, I will discuss the ethical considerations that are involved with measuring software engineering and the effect that this can have.

1.2 – What is Software Engineering?

Software engineering is a computing discipline associated with development of software product. Software engineers use well-defined scientific principles, methods and procedures to create, modify and improve software.

The field of software engineering is of huge importance today. In a world where most industries rely on computers and databases, there is a constant demand for new software as large companies often rely on computer systems to turn a profit. The IT industry as a whole can be used as an example. Companies like Facebook, Google. Microsoft, Amazon and Apple are constantly searching for the top software engineers in the world to help develop the software not only for their products, but also to help their company run seamlessly and efficiently.

1.3 – Why do we measure Software Engineering?

Like all other disciplines, software engineers wish to work in the fastest and most efficient way possible. By tracking, analysing and measuring the process of software engineering, companies can gather data on their employees as they work on the software. The goal of tracking and analyzing software engineering like this is to assess the level of quality of the current product or process, improve that quality and predict the quality once the project has been completed. This data can be used to identify areas of low-productivity aswell as areas of high-productivity, both of which can be assessed to form plans on how to maximise efficiency and productivity. It can also be used to manage workloads, reduce overtime and reduce costs.

1.4 – What do we measure?

Meausring software engineering involves the use of software metrics. A software metric is a measure of software characteristics which are quantifiable or countable. These software metrics can be used for many purposes, such as measuring software performance, planning work items, measuring productivity, and many other uses.

It is extremely difficult to say which particular software metrics are most important when it comes to measuring software engineering. Different companies will have different engineers working on different projects for different goals. So when measuring software engineering, companies will never focus on the same software metrics. Here are some examples of software metrics that companies use when measuring software engineering:

- Sprint burndown
A burndown report shows how the workload of the sprint was distributed and completed based on story points.

- Team velocity metric
Shows the amount of software your team completed during a sprint.

- Throughput
Indicates the total value-added work that was done by the team.

- Cycle time
The time it takes to complete a task in full.

- Lead time
The time it takes from when a new feature is defined to the user to when the feature is available to the user.

- Mean time to repair
How quickly fixes can be deployed to the user.

- Code coverage
The quantity of code, measured in lines of code, that is covered by a unit test.

- Bug rates

The average number of bugs that are created when a new feature is deployed into the code.

- Task volume + average estimates
The amount of tasks that your team can overcome when faced with change, compared against the average estimates is a great way of understanding how consistently a team is completing their work.

- Recidivism
A good indicator of incomplete or inconsistent requirements that someone in a team may want to investigate to see who is not putting in enough work.

- Amount of commits
The number of times someone has committed work to a repository conatining the software/project.

- Frequency of commits
How frequently someone is committing work to the repository.

1.5 – How do we assess the data?

Assessing the data gathered by tracking these software metrics helps highlight what factors are of benefit to a team, and what are holding them back. We can compare the costs and benefits of certain practices to work out which one is worth the cost of performing it. Later on in this report I will describe the different services that companies are offering when it comes to measuring software engineering, and how they go about tracking and analysing these software metrics.

2 – Analytical Platforms

2.1 – Pluralsight

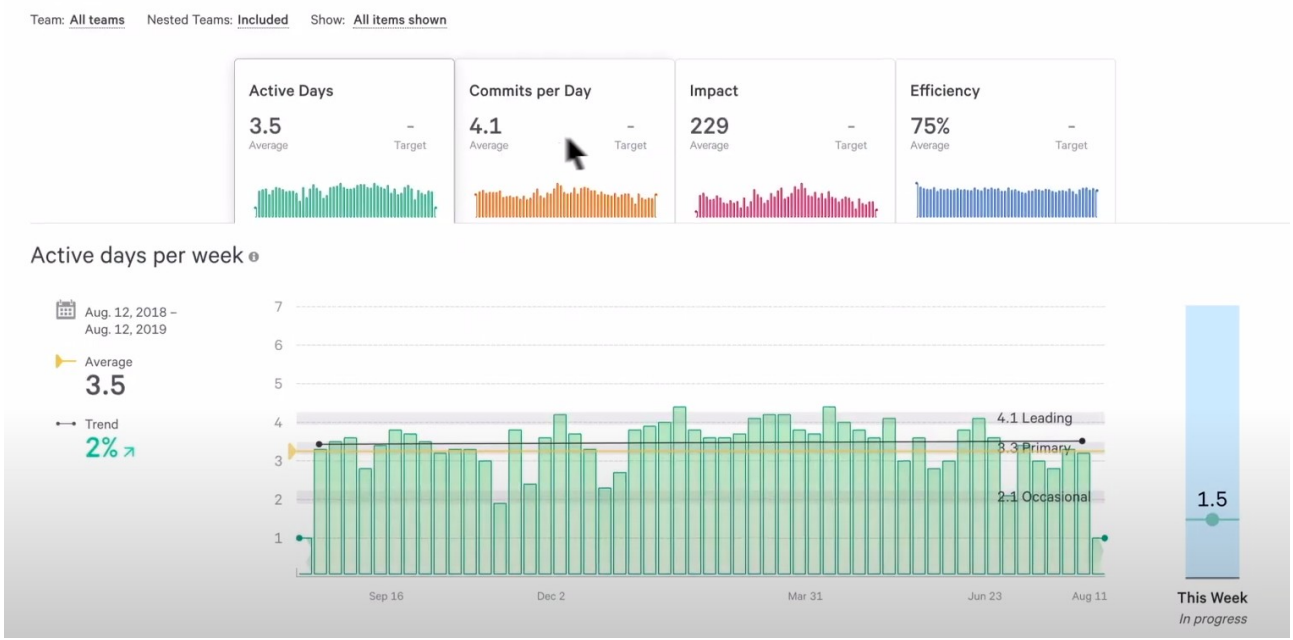
The company Pluralsight offer a service called Flow that, in their words, “provides engineering leaders with metrics in context to ask better questions and advocate for the team with substantive data”. This service was formerly known as GitPrime until Pluralsight acquired it and took over in 2019.

By mining data from Git, Flow provides contextual software metrics that the engineering leaders can assess and utilise to analyse their teams and make changes based off the substantive data that they gather.

Flow is designed specifically for leaders of software development teams. By focusing on the anlysis of source control, specifically Git, Flow allows these leaders to gain insights from the study of these repos and ultimately measure their team's success. Some things that Flow could offer are metrics such as the amount of the team's burn that is allocated to refactoring old code, how specififc meetings are effecting productivity, how well knowledge is shared in code reviews or what certain sectors of the team accomplished in a week.

The data gathered by Flow is very well presented and easy to understand. Flow takes data from code commits, pull requests and tickets that offers the team leaders an insight into their software development process. This enables the leaders to manage their teams in a much more efficient manner, and also allows them to utilise their team members more efficiently.

The data is presented in many different ways. A detailed view into the team's workflow is provided through a series of graphs and charts that are littered with important information on each team member . Team leaders can use this data to encourage healthy commits, and also spot roadblocks in the code by viewing open pull requests and unresolved comments. All of this can be extremely helpful for anybody that leads a software development team.

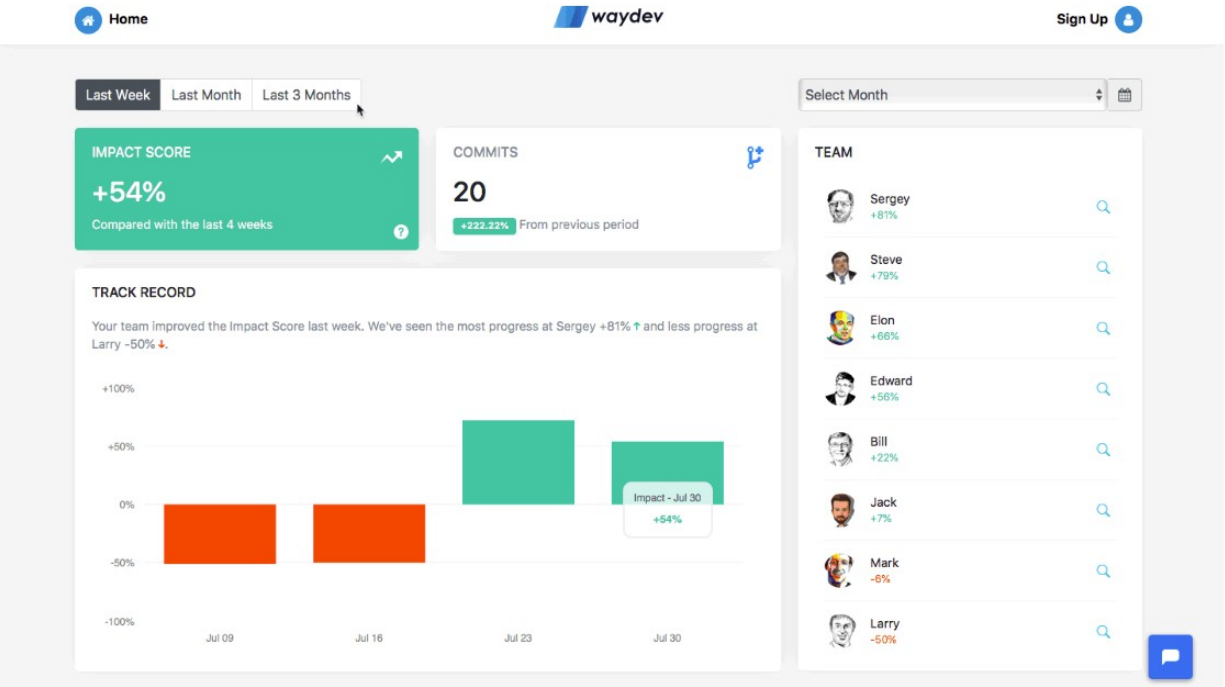


Flow Interface

2.2 – Waydev

Waydev is another great example of a Git analytic platform that can be used to track software metrics and help software development leaders push their teams and excel in the industry. Waydev is an agile data-driven platform that tracks software engineers output straight from the Git repo. They gather their data from various different sources, such as Github, Gitlab, Azure DevOps and BitBucket.

Much like Flow by Pluralsight, Waydev is designed for the leaders of software development teams that wish to bring out the best in their engineers. Waydev analyses the team's code base, pull requests and tickets and using this data it provides easy to understand insights and reports that increase the leader's understanding of the team and improve efficiency and velocity of the team as they progress through the software development project.

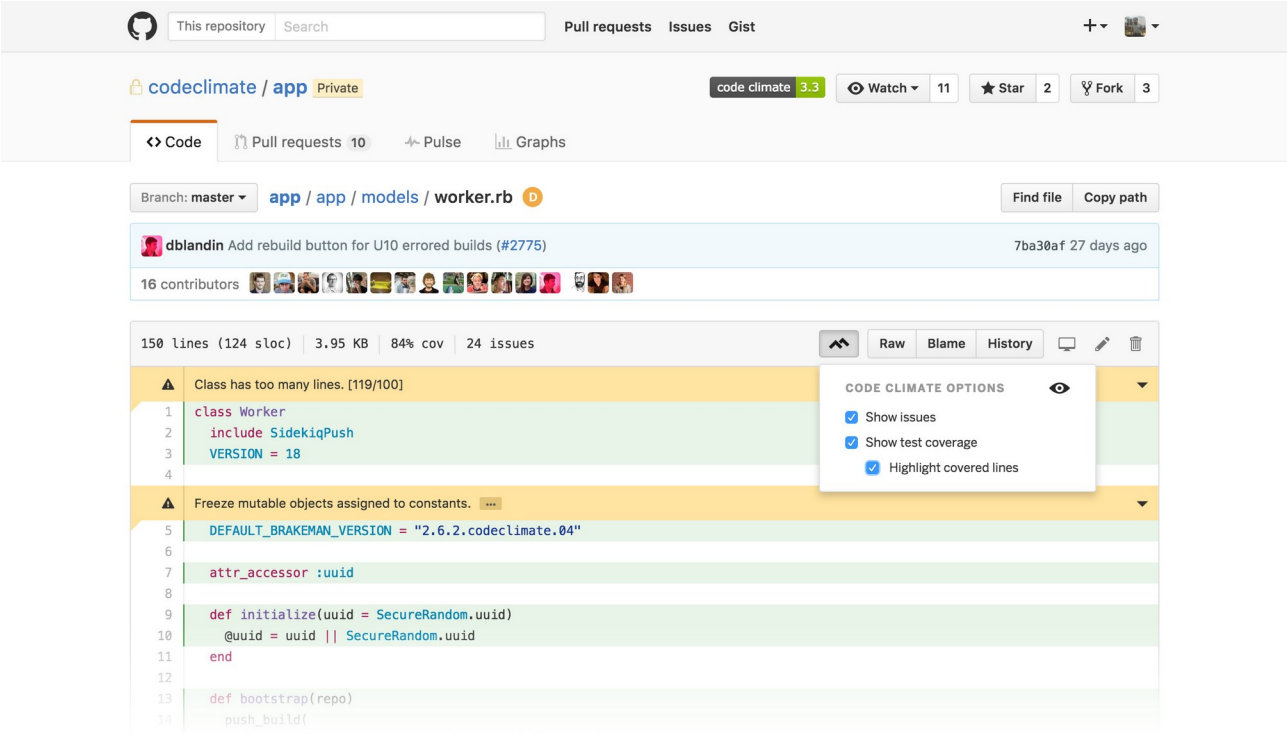


Waydev interface

2.3 – Code Climate

Similar to both Pluralsight and Waydev's approach to measuring software metrics, Code Climate gathers data from commits and pull requests and aims to provide insights into a team's productivity and help software development team leaders make lasting improvements to their team and how they function.'

Code Climate integrates directly into the repo containing the software/code and offers intsant insights into the software metrics of the project. It gathers and assesses data from commits, pull requests and can also pinpoint code that could have a negative effect on the project such as code that has been copy and pasted from other sources.



GitHub interface using Code Climate

3 – What algorithms can we use?

3.1 – Using machine learning

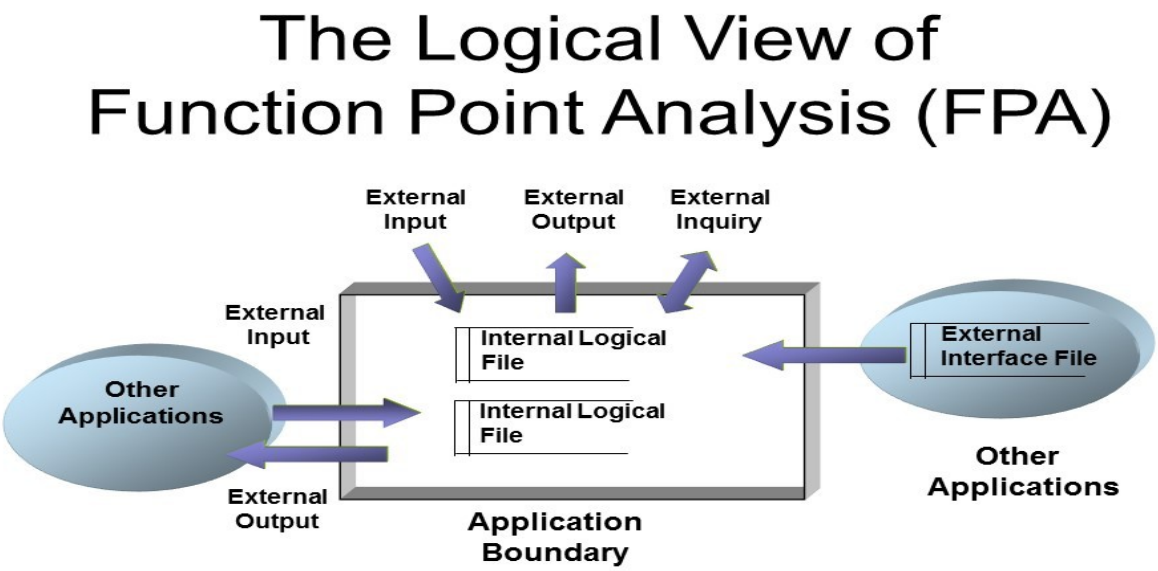
One tool that we can use when measuring software engineering is using machine learning to generate software metrics for fault prediction.

Testing software is vital when it comes to writing good software, but it can be extremely time consuming and tedious. Through the use of machine learinng we can generate techniques that predict the testing effort of software that is being written. This significantly increases time-efficiency, effort and cost usage. Over the last ten years, this has become a hot topic in the industry and considering the vast developments when it comes to machine learning it is no surprise that there are many great examples of how machine learining can be used to identify faulty modules in software today.

3.2 – Functional point analysis

Functional Point Analysis is a method that is standardized to measure a software work product. FPA is a technique that is utilised to measure software requirements based off of the different functions that the requirement can be divided into. It was initially developed by Allan J. Albercht in 1979 at IBM. Since it’s initial release it has been further modified by the International Function Point Users Group (IFPUG).

FPA is used to measure the amount of functions that a software provides to the user. By keeping track of this number, we can measure the overall progress of software development. FPA can also be used to precisely estimate the time and cost that is required to implement a specific amount of functions by using data that was previously gathered.



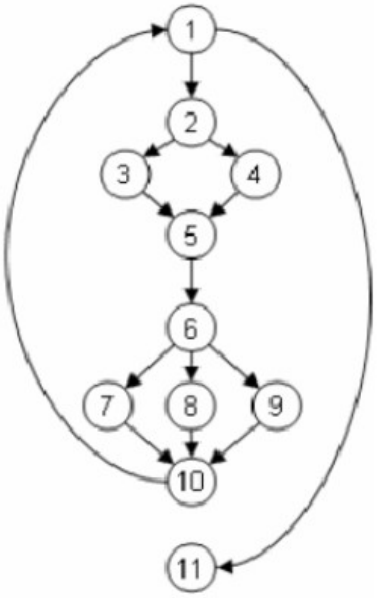
Functional point analysis diagram

3.3 – Cyclomatic Complexity

Cyclomatic complexity is a software metric that can be used to calculate the complexity of a computer program. The number of linearly independent paths through a program are calculated from the source code of the software in question. T compute cyclomatic complexity, the control flow graph of the program is used. To put it simply, the higher the count of linearly independent paths tha are in the source code, the more complex the software is.

Cyclomatic complexity can be used in tow ways, either to limit the complexity of code, or to calculate the total number of test cases that are required. Cyclomatic complexity is by far one of the most difficult and comlex software metrics to understand and therefore is extremely hard to calculate. Although it is very hard to calculate, it can be extremely useful for team leaders that wish to gain a better understanding of the software that they are developing.

Node	Statement
(1)	`while(x<100){`
(2)	`  if (a[x] % 2 == 0) {`
(3)	`      parity =  0;`
	`   }`
	`   else {`
(4)	`      parity =  1;`
(5)	`   }`
(6)	`   switch(parity){`
	`      case 0:`
(7)	`        println( "a[" + i + "] is even");`
	`      case 1:`
(8)	`        println( "a[" + i + "] is odd");`
	`      default:`
(9)	`        println( "Unexpected error");`
	`    }`
(10)	`   x++;`
	`}`
(11)	`p = true;`



Control flow graph of a program

4 – Ethical Concerns

4.1 – Misuse of software metrics

Problems with measuring software engineering could arise if a narrow selection of software metrics are chosen when measuring software. Some developers could be labelled as unproductive when in reality the software metrics that are being analysed just may not highlight the work that the developer is actually doing. For example, if a team leader only focuses on the amount of code written, developers that write more code might be rewarded or recognised over other developers that have not written as much code. This gives an inaccurate representation of who is doing more software engineering as one could be writing an excess of mediocre code whereas the other could be writing small amounts of really good code that are of huge importance to the software. This is why I think that it is important that team leaders use a wide variety of software metrics when analysing their teams, and I think that this is why the platforms like Pluralsight, Waydev and Code Climate can be extremely benefitial to teams as they offer a vast range of software metrics and gather a lot of important data that would give an accurate representation of the work that is actually being done by a developer in a team.

4.2 – Data privacy

Another problem that could arise with measuring software engineering is a very common problem in today's world and that is data privacy. When collecting data on teams and the developers within these teams, it is possible that some of the data that is gathered could violate a devloper's data privacy rights so it is extremely important that the necessary precautions are taken when measuring a software engineers progress and gathernig data on them as they work. It is important to outline to the developer what data is being gathered and to make sure that they consent to this and are fully aware of the process.

5 – Sources

https://www.youtube.com/watch?v=OkPF27xW5kk

https://waydev.co/about-us/

https://www.youtube.com/watch?v=Oj_g7N2KY-Y

https://www.youtube.com/watch?v=r1joerjQL_c

https://www.fingent.com/blog/function-point-analysis-

introduction-and-fundamentals/#:~:text=What%20is%20Function%20Point%20Analysis,summarized%20using%20the%20FPA%20formula.

https://www.perforce.com/blog/qac/what-cyclomatic-complexity
https://www.researchgate.net/publication/323719116_Software_metrics_for_fault_prediction_using_machine_learning_approaches_A_literature_review_with_PROMISE_repository_dataset