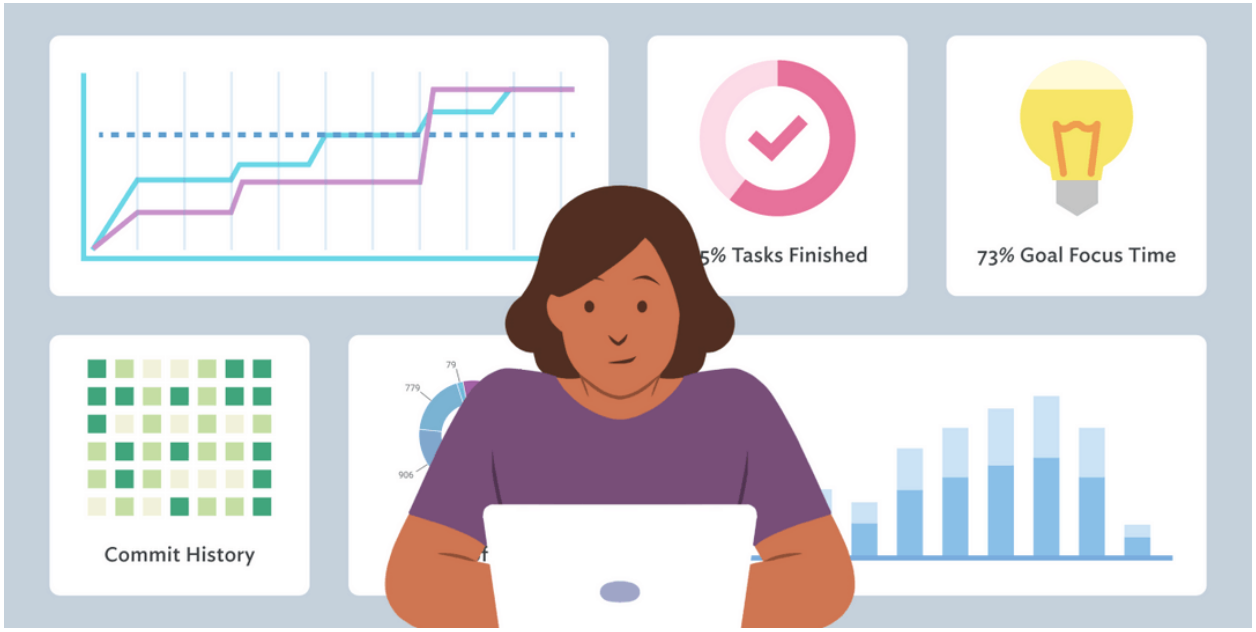
CSU33012 SOFTWARE ENGINEERING

Measuring Software Engineering Report



John Kommala, Std# 19303445

Contents

1. **Introduction**

2. **How can be Software Engineering Measured**
   - 2.1. Reasons behind measuring
   - 2.2. Methods of measuring
     - 2.2.1. Lines of code
     - 2.2.2. Number of code commits
     - 2.2.3. Code Review

3. **Platforms for Measuring Software Engineering**
   - 3.1. SonarQube
   - 3.2. Pluralsight Flow (GitPrime)
   - 3.3. Velocity 2.0
   - 3.4. Jira

4. **Computations over Software Engineering Data**

5. **Ethical Concerns**

6. **Conclusion**

7. **References**

1 Introduction

This report will focus on measuring software engineering based on aspects like how one can measure software engineering activity, the platforms that enable us to measure this, and the kind of computations that could be done over software engineering data to profile the performance of software engineers. These aspects are outlined considering ethics and legal or moral issues surrounding the processing of such data.

2 How can be Software Engineering Measured

In order to discuss how software engineering can be measured, let us look at the reasons behind measuring software engineering.

2.1 Reasons behind measuring

There are many advantages of measuring software engineering, for example, measuring software engineering can result in an efficient development environment which could increase the quality of the code/software produced, enabling the maintenance aspect of the development process. This would also contribute towards decreasing the cost of the development and maintenance of the software and potentially result in a faster development process.

2.2 Methods of measuring

There are countless ways of measuring software engineering activity from a software development team ranging from calculating the complexity of code by looking at its structure to estimating the productivity of a software engineer based on their code commit frequency.

2.2.1 Lines of Code

While measuring software engineering, the simplest metric towards this process is considered to be counting the lines of code. This is a simple and crude way of estimating the productivity of a software engineer. However, using this metric has resulted in encouraging bad code practices and less efficient code. For example, a developer may

have produced over 10,000 lines of code but we can't be sure that these lines are efficient and free of bugs(errors). Furthermore consider the example below, comparing the first code snippet to the next one, we can see that the first snippet is almost 5 times more in terms of lines of code, but does exactly the same thing. This proves that counting lines of code is not an accurate way of measuring the productivity of a developer based on lines of code.

Figure 1: Sample code with more lines due to syntax

```
count = count < 5 ? count + 1 : count - 1;
```

Figure 2: Code with same function as Figure 1, but in less lines due to a different syntax

2.2.2 Number of Code Commits

Usually when a group of developers are working together on a project, pull requests are generally considered a better practice to make changes to the code base. A pull request is a way of letting the rest of the team know about changes a developer has pushed on to a branch. All the commits a developer would make would be counted under this branch which would be under the pull request. Generally, the more pull requests a team completes, the more productive they are considered to be. But this is not an accurate way of measuring productivity either. This is similar to counting lines, which encourages bad practices like making vague changes just for the sake of it encouraging small and unnecessary pull requests.

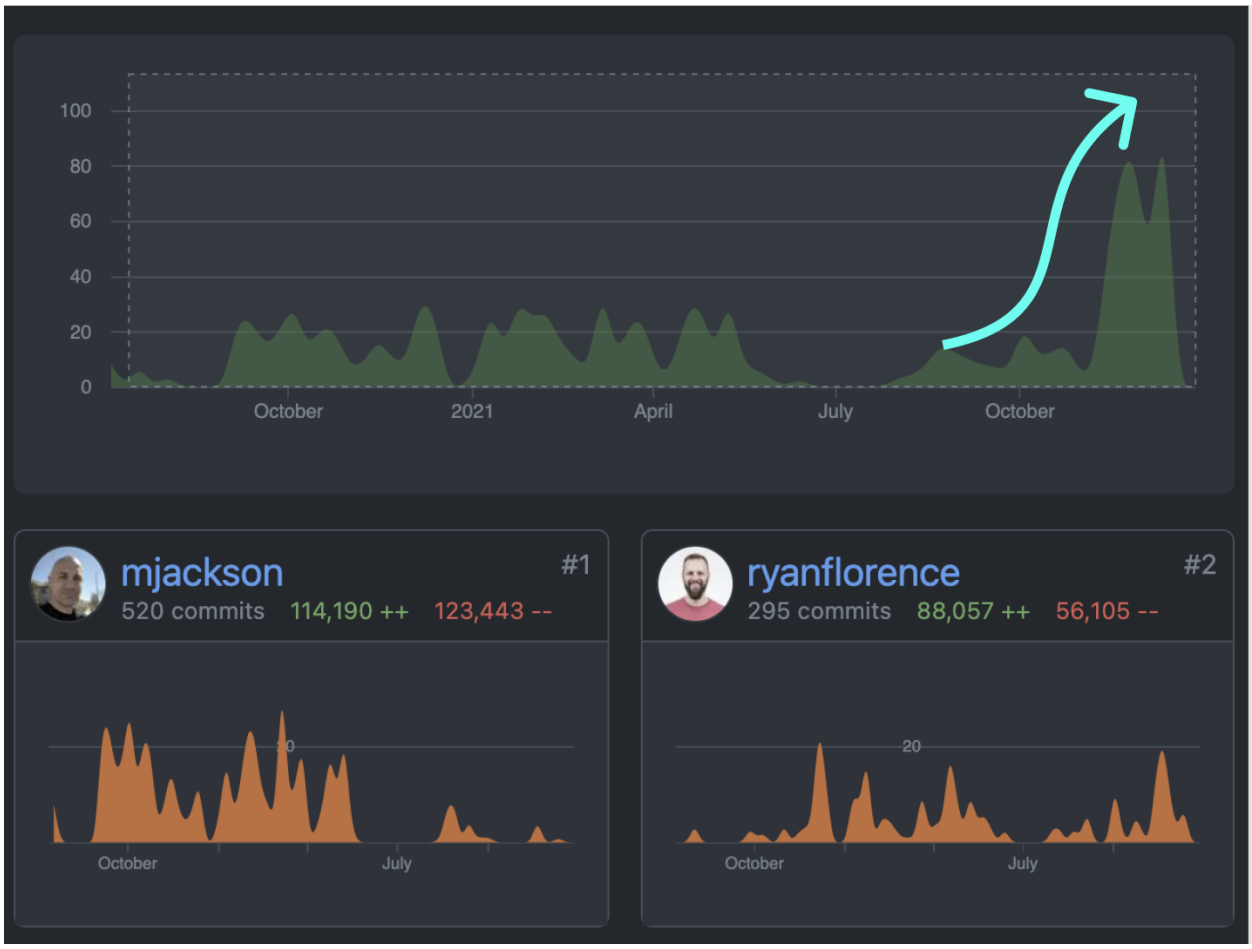


Figure 3: Data related to commits on github on an open source project

2.2.3 Code Review

Code Review is a modern approach on measuring software engineering. Code Review, also referred to as peer review, is a software quality assurance activity in which one or several people check a program mainly by viewing and reading parts of its source code. In many teams senior developers are often the ones reviewing code for their subordinates. This is considered as a better method of measuring software engineering as it prioritizes process metrics over product metrics. For example, whenever some new feature is implemented or new code is added or existing code is modified/optimised, a code review is requested. This usually involves peers and more experienced developers would look over the changes and make any suggestions before merging the code into the main code base. The productivity is measured based on the time it takes to finish a code review and merge the changes.

3 Platforms for Measuring Software Engineering

As measuring software engineering is considered valuable by companies, many services and platforms have been created to help companies analyze and visualize these metrics. The information provided by these services help employers understand and visualize the current status of a project and estimate the time it will take to finish the project,etc. Some of these services even enable the companies to identify the employees that have a high impact on the company.

3.1 SonarQube

SonarQube is a free and open source platform that is used to inspect the code quality continually within a code base. The language support is limited to about 29 languages, covering most of the popular languages used in the industry. It checks for bugs, redundant code, provides code coverage and some other issues within the code base. This data is also analysed to help the developer optimise their code.

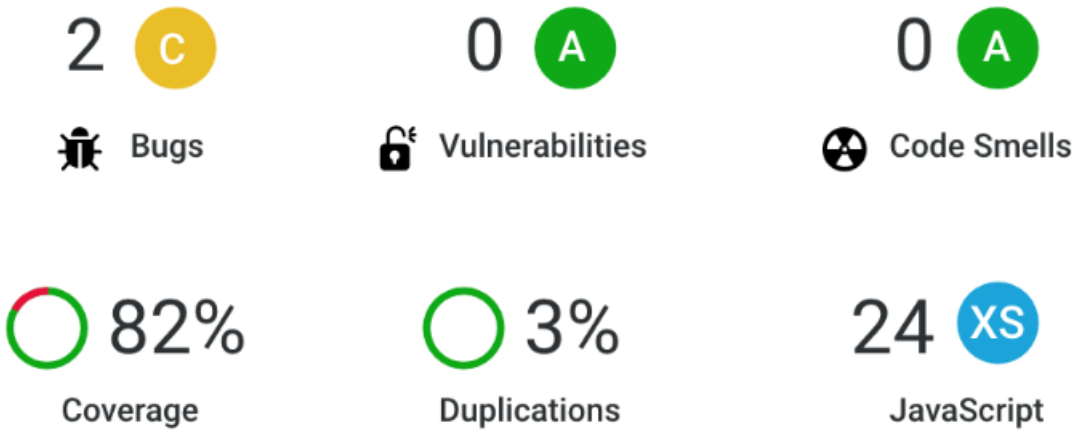


Figure 4: Some process metrics from SonarQube

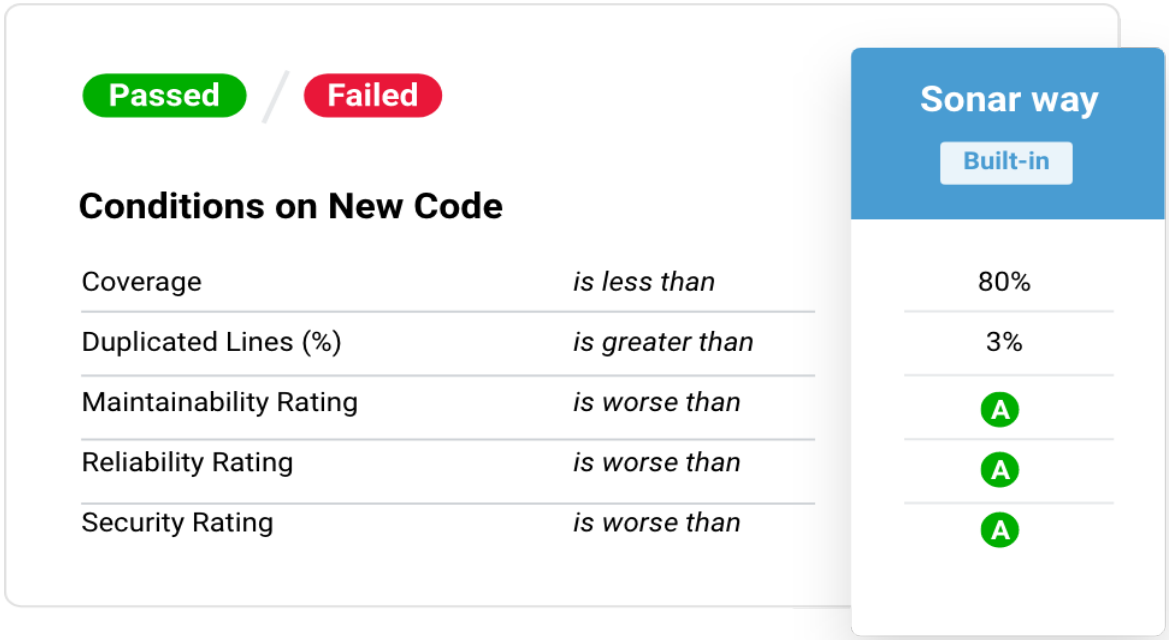


Figure 5: Some unique relative metrics from SonarQube

3.2 Pluralsight Flow (GitPrime)

Pluralsight Flow, previously known as GitPrime is a service which fetches data from a git host (like Github, Gitlab,etc) and the git information from the required repositories. This data is processed and used to visualize useful metrics to estimate the performance of software developers. Version Control Systems like git store a large data footprint for the developers that use it. This generally consists of the quantity of commits, lines changed, etc. As mentioned above, these metrics on their own are not proper indicators of productivity or efficiency but when we put many or all of these metrics together, we can eliminate some of the flaws and get a better estimation of the productivity. One of the advantages of Pluralsight Flow is it uses the information from pull requests linked to commits which would produce some insightful graphs.

Figure 6: Infographics from PluralSight Flow based on certain parameters

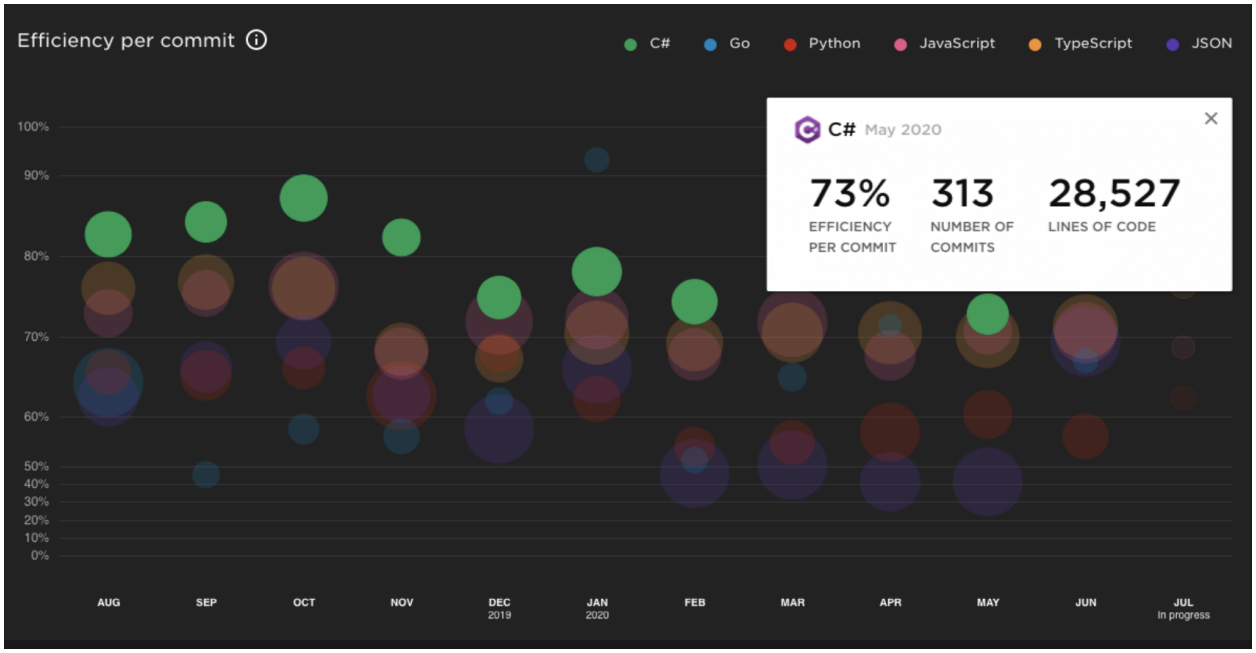


Figure 7: Infographics from PluralSight Flow based on languages and efficiency

3.3 Velocity 2.0

Velocity 2.0 by Code Climate, helps analyse data from a repo focusing more on individual developers in a team. This makes it different from PluralSight Flow, as it is aimed towards groups of developers as a whole. This is also aimed towards measuring process metrics over product metrics. It also visualises raw engineering/ contribution data that would be helpful for someone without a software engineering background.

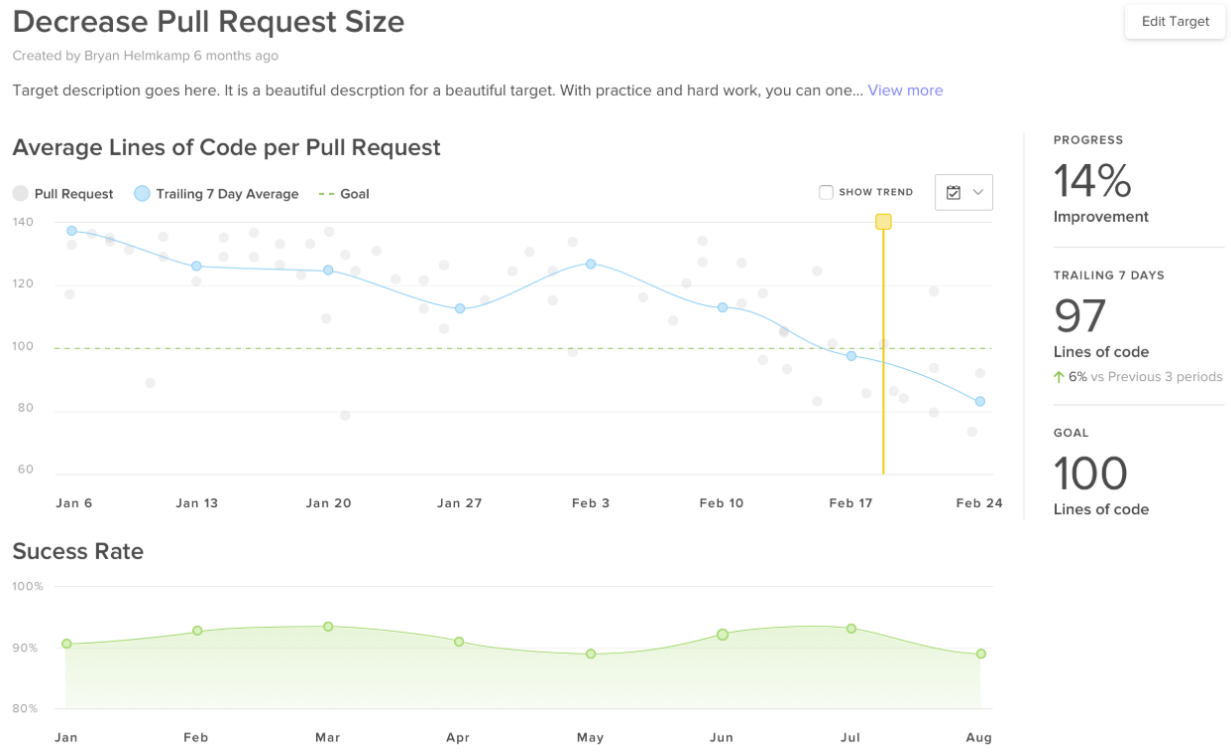


Figure 10: Infographics from Velocity 2.0

3.4 Jira

Jira by Atlassian, has been a popular tool amongst software engineering teams. It is mainly focused on teams that use agile methods. Jira provides some useful infographics like roadmaps, scrum boards, sprint reports, etc.

4 Computations over Software Engineering Data

While looking at various kinds of computations that could be performed over software engineering data, we have a popular computational approach that involves a regression model based on the number of lines of code called Constructive Cost Model. This is a procedural cost estimate model for software engineering projects which is often used as a process of predicting parameters like size, effort, cost, time, etc associated with the project. Projects that use this model are classified into three categories, organic, semi-detached and embedded. An Organic Project would be around 2,000 to 50,000 lines of code, belonging to a small team with good domain knowledge and familiar environment. A Semi-Detached Project would be around 50,000 to 300,000 lines of code, with a medium team with a mix of experienced and inexperienced developers to deal with less familiar environments. An Embedded Project would be above 300,000 lines of code, with experienced developers dealing with completely new and unfamiliar environments.

Machine learning is used extensively nowadays to analyse large datasets as it has been proven to be quite useful, we can see this being used in the real world with issues like medical diagnosis, image recognition, predictive analytics, etc. Machine Learning is divided into three types based on the way a model is trained, they are Supervised learning, Unsupervised learning, and Reinforcement learning. Firstly, Supervised learning is when a model is trained with a dataset that is labeled. For example, consider a large dataset of chest x-rays that are classified into positive and negative when checked for pneumonia. After training the model, when we run a new chest x-ray through the model it will tell us if the x-ray is positive or negative with a certain accuracy rate. Unsupervised learning on the other hand is when we have unlabelled datasets, the model under this training would come up with correlating factors on its own through exploration. Recommendations on apps like youtube and some online shopping websites is a good example of unsupervised learning. Looking at Reinforcement learning, it is similar to unsupervised learning, as the data is not labelled, but the dataset here has a boolean feedback element associated with it. An example of this would be path-finding algorithms. These methods can be used to analyse and estimate the efficiency and productivity of a developer in the future using related data.

5 Ethical Concerns

Based on my research for this report, I find that measuring software engineering from a management perspective would be very important and there would definitely be some ethical concerns related to this.

Firstly gathering data based on code should be of very little ethical concern as this would be something the developer would be working towards and the company would hold the rights to the software and code base as a whole.

However there might be some concerns about analysing the data, this comes down to the techniques used to analyse this data. Sometimes there might be a chance of producing incorrect results on the data from the employees which could lead to misinterpretations. In some cases where there are third parties services and platforms involved in analyzing this data, the management or the person in charge should make sure that the employees data is not being risked. There would be little to no chance for the company to look at the internal processing and working of some third-party services, which would make it quite hard to even identify when incorrect results are produced. Keeping these situations in mind It would be reasonable to question the accuracy of the results produced using this data and other related metrics. Next comes interpretation, sometimes the results and infographics produced using the developers metrics could be interpreted in different ways depending on their perspective and thinking. It could sometimes be presented in misleading ways. These situations could result in great misunderstandings, not valuing the employee's work enough could lead to producing less quality work and sometimes it could even cost the employee their job depending on how bad the misunderstanding is.

6 Conclusion

In Conclusion measuring software engineering is complex, however there are many ways to do so using relevant metrics and tools mentioned previously in the report. These services and metrics could enable developers to improve their productivity and identify areas and scope for improvement.

7 References

1 Introduction

Picture - https://www.getclockwise.com/blog

2 How can be Software Engineering Measured

https://www.getclockwise.com/blog/measure-productivity-development

https://en.wikipedia.org/wiki/Software_metric#Common_software_measurements

https://www.geeksforgeeks.org/measuring-software-quality-using-quality-metrics/

3 Platforms for Measuring Software Engineering

https://www.sonarqube.org/

https://www.pluralsight.com/product/flow

https://codeclimate.com/velocity/

https://www.atlassian.com/software/jira

https://docs.github.com/en/enterprise-server@3.0/admin/overview/system-overview

4 Computations over Software Engineering Data

https://www.educba.com/cocomo-model/

https://www.geeksforgeeks.org/software-engineering-cocomo-model/

https://www.researchgate.net/publication/237050541_Measuring_productivity_of_software_development_teams - Sudhakar, Goparaju & Farooq, Ayesha & Patnaik, Sanghamitra. (2012). Measuring productivity of software development teams

5 Ethical Concerns

https://www.google.com/about/datacenters/data-security/

https://www.researchgate.net/publication/3507312_Ethical_considerations_in_software_engineering - Gotterbarn, Donald. (1991). Ethical considerations in software engineering. 266 - 274. 10.1109/ICSE.1991.130652.

https://www.researchgate.net/publication/298902502_Software_Security_Privacy_and_Dependability_Metrics_and_Measurement - G. Hatzivasilis, I. Papaefstathiou and C. Manifavas, "Software Security, Privacy, and Dependability: Metrics and Measurement," in IEEE Software