**Measuring Software Engineering**

“To deliver a report that considers the ways in which the software engineering process can be measured and assessed in terms of measurable data, an overview of the computational platforms available to perform this work, the algorithmic approaches available, and the ethics concerns surrounding this kind of analytics.”

[**Measurable Data**](#_plk9r7jlu9zs) **2**

[Number of Commits](#_9psfivrvqrmv) 2

[Technical Debt](#_k7s2ccq2svly) 2

[Communication](#_b8kz6kjmj6o8) 3

[**Computational Platforms**](#_ha0fqitu8tr) **4**

[Analytics as a Service](#_vflpskocqibo) 4

[Code Review Platforms](#_s9sh0gj87048) 4

[Code Climate](#_x44l9sm2z4fh) 4

[Codacy](#_t1xpm03fa2rw) 5

[CodeBeat](#_qxc24ds66rmb) 5

[**Algorithmic approaches**](#_st77ah83qre8) **6**

[Supervised Learning](#_z325z2jkma0p) 6

[Computational Intelligence](#_56ws431udolb) 7

[**Ethics**](#_ihgdjiyo73tx) **8**

[Ethics in Software Engineering](#_xow33vlno1tw) 8

[How can Software Engineers Harm the public?](#_hsx5q4b6p15k) 8

[How can Software Engineers Bring Good to the Public?](#_nuwrizdig7ux) 9

[Beneficial or Detrimental?](#_ax9171h00yx) 9

[**Bibliography**](#_8a60mnjnt6d8) **10**

# 

# 

# Measurable Data

## Number of Commits

Software engineers are often tracked on the number of commits they perform in a given period of time. Most people argue that this is a poor way to measure software engineers. They say that if engineers are being measured this way they will just reduce the number of lines of code per commit and commit more often. This will lead to more complicated commit histories and more frequently merging of code. Also, a high number of commits tells you nothing about the quality of the code being written.

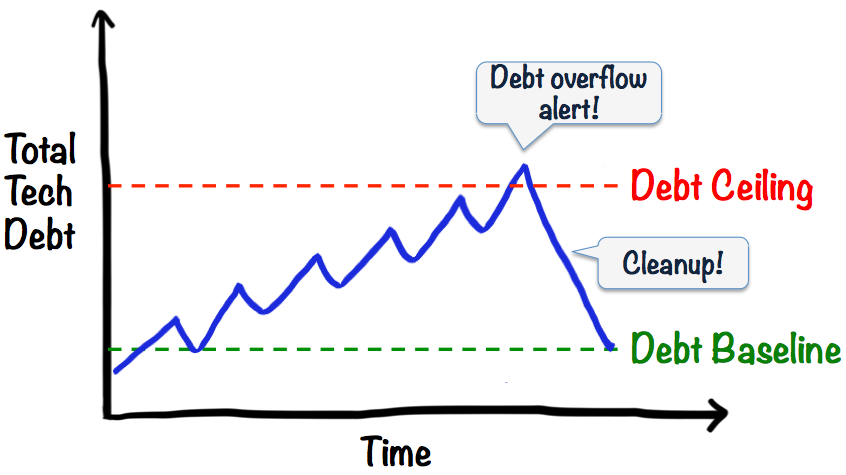
However, some people argue that tracking the number of commits can give them quality insights into their software engineers. “I would go to every active repository and manually write down the number of code commits from my developers.” Brian York Technical Entrepreneur. He states that more often than not a drop off in commits was a good indicator that something was going on with the developer.

Although the number of commits does not tell us the full picture I would take the opinion that it is a very important metric that cannot be ignored when analysing a software engineer’s productivity.

## Technical Debt

Technical debt is a concept in software engineering that reflects the extra development work that arises when code that is easy to implement in the short run is used instead of applying the best overall solution. A little debt speeds development so long as it is paid back promptly with a rewrite. The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt.

The debt ceiling should be set high enough that we don't hit it very often but kept low enough that we aren’t ultimately screwed when we do hit it. An inappropriately high debt ceiling can be disastrous to large software projects. “Entire engineering organizations can be brought to a stand-still under the debt load of an unconsolidated implementation, object-oriented or otherwise."

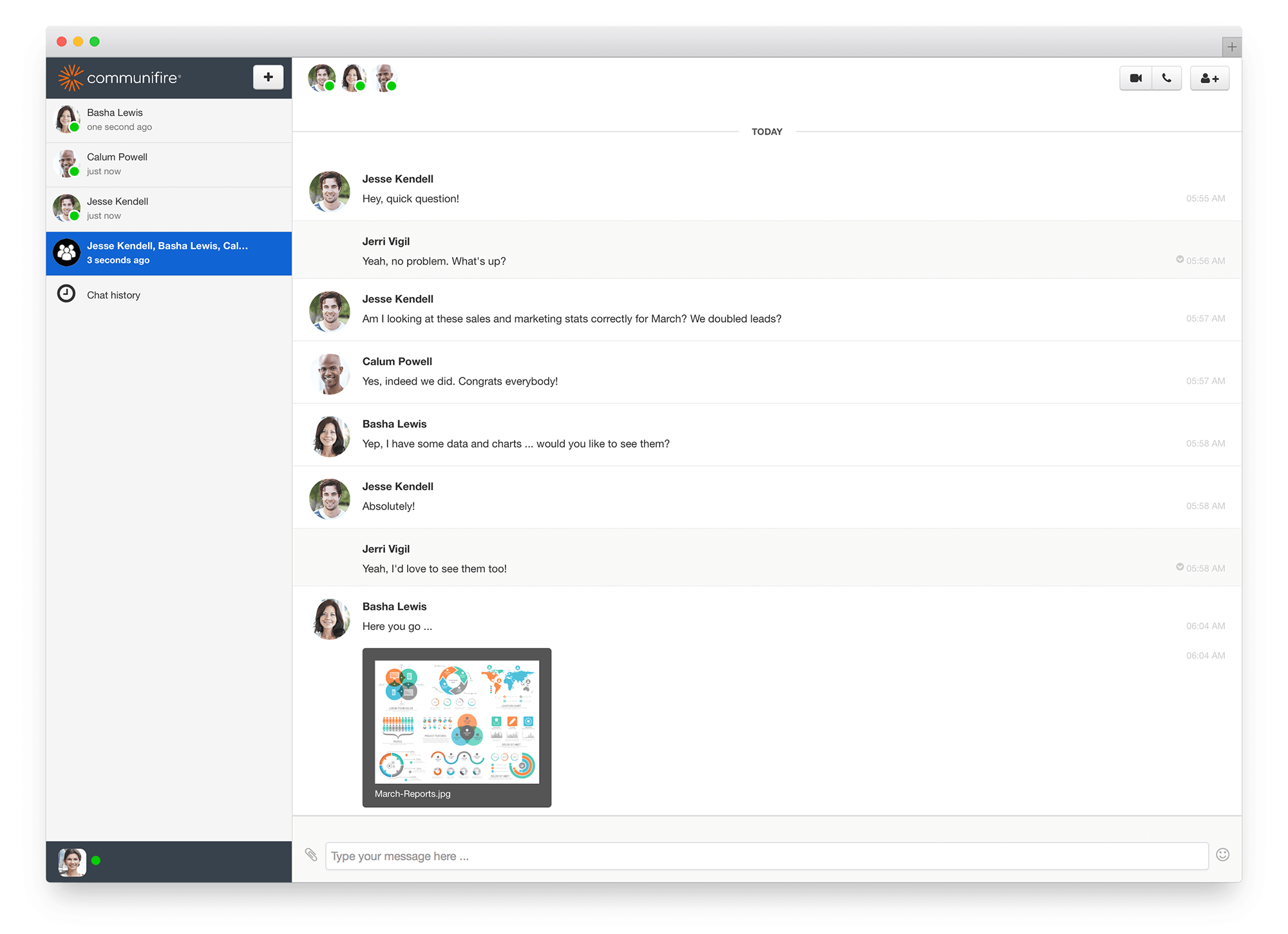


It is important to note that technical debt is not poor code. Technical debt can often be well written code coming from a good software engineer under unrealistic time constraints. Therefore, an understanding relationship between management and the software engineers is crucial to keep the technical debt at a sustainable level. This point will flow into my next point about communication. Communication is essential when ensuring the technical debt does not break its debt ceiling.

## Communication

In my opinion communication is up there with the most important things that should be measured as a software engineer. “If you possess the ability to work well with others, to be able to lead and to follow when required and to communicate your ideas clearly and empathetically, then you possess something that is far more valuable than programming knowledge or experience. “ John Cairns Software developer. I would have to agree with John here. If a software developer is struggling to submit lines of code he can put some extra work in, gain greater knowledge of the task at a hand and he will more than likely complete his task. However if you have a software engineering that is unable to communicate with his team members or his manager that may severely hamper him and his team in the long run.

Due to the fact that I value communication so highly I would see fit that companies should install an Intranet Chat for their employees to communicate on. This would allow for a decent measurement of how often and how effectively employees communicate over their computers. However, to fully assess how well employees communicate we must measure how they interact in person. One way of doing this would be to run peer reviews where employees would rate their co-workers ability to communicate. This would not be 100 percent effective as employees may be kind to the people they like in the office, however I do believe you would get some useful insights.



# Computational Platforms

In this section I will discuss the growing trend of analytics as a service and compare the features of some code review platforms available today.

## Analytics as a Service

Analytics as a service (AaaS) refers to the provision of analytics software and operations through web-delivered technologies. These types of solutions offer businesses an alternative to developing internal hardware setups just to perform business analytics. Analytics as a service is becoming a valuable option for businesses because setting up analytics processes can be a work-intensive process. Businesses that need to do more analytics may need more servers and other kinds of hardware. They may also need more IT staff to implement and maintain these programs. If the business can use analytics as a service instead, it may be able to bypass these new costs and new business process requirements. This is especially true for SME’s with a limited budget and a small number of staff available to them.

## Code Review Platforms

I will now compare some of the major code review platforms available on the market today and detail what they offer. The three platforms I will examine are code Climate, Codacy and codeBeat.

### Code Climate

* Huge number of supported languages, technologies and frameworks eg Java, PHP, Ruby on Rails
* Used by big players in industry such as Pivotal, JQuery for open-source, Salesforce and Intercom
* It has a really nice new UI
* “Quick wins” list as a feature
* Trend charts showing the development as trends such as technical debt (see picture below)
* However it is the most expensive on the market

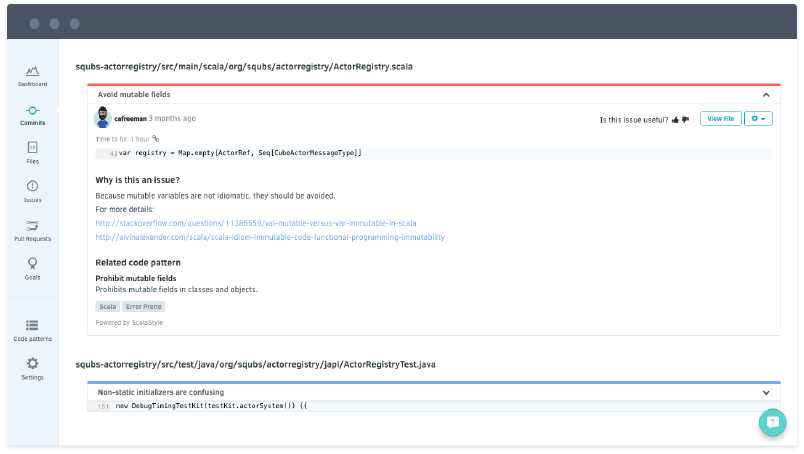


(Example of a trend chart for the technical debt in the project. See how it shows the estimated remediation time.)

### 

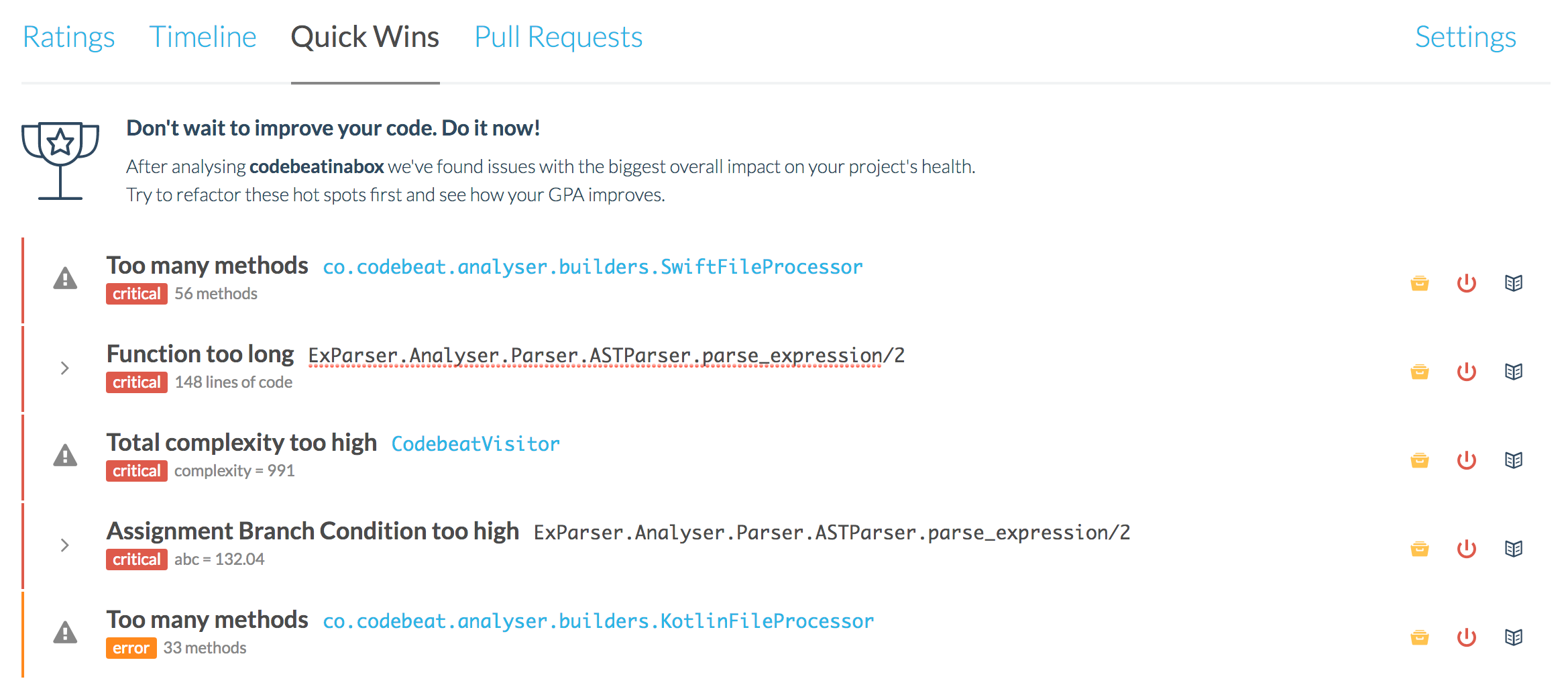
### Codacy

* Used by some of the biggest firms in industry eg. Paypal
* Really nice feature of estimating the expected time to fix issues
* You can track your project quality evolution with a friendly UI dashboard
* You can create your own code patterns that Codacy will run against the code for detailed code analysis
* Codacy informs software engineers of the company coding guidelines and gives them well-described issues with examples below each case (see picture below)
* Sends you notifications on the issues you deem important - weekly email etc.
* However no “quick wins” feature
* It does not have the capability to search for issues



### CodeBeat

* Excellent customer support from their team
* Customer suggestions are considered and implemented in the product if deemed useful (the other platforms don’t offer this)
* Small but well documented API
* Unique quick wins tab (nice feature see picture below)
* It is a small company that is dynamically growing
* Missing some things in documentation eg. code duplication
* No possible security issue checks (most others offer this)
* No CSS analysis



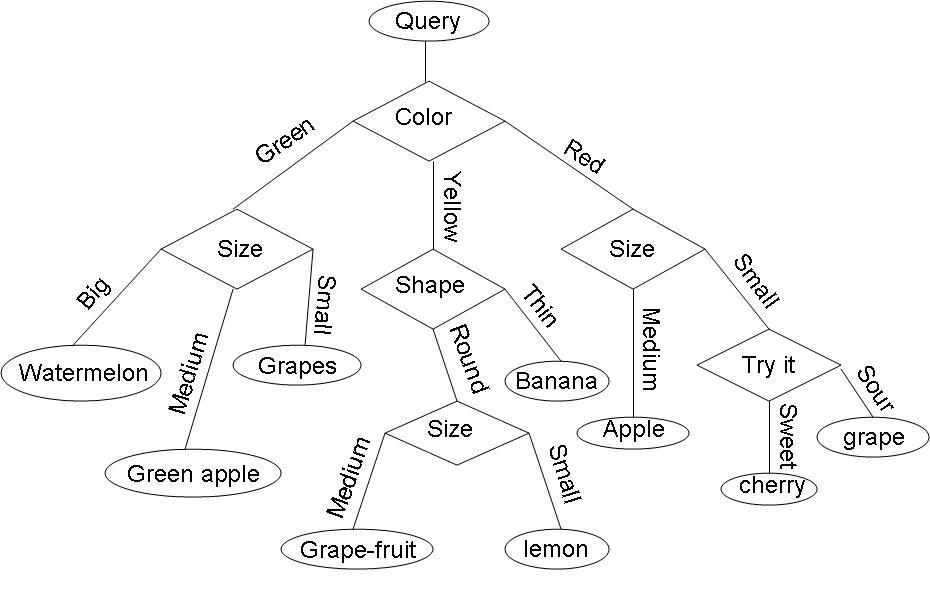
As you can see all three platforms offer many valuable functions. However the one that impresses me the most is Codacy. The main reason being their detailed, extremely helpful issue resolving messages. They inform the engineering of the company coding standards and give them examples on how to go about fixing the issue. It is also the cheapest of the three at $15 per month per user.

# Algorithmic approaches

## Supervised Learning

Supervised learning is the machine learning task of inferring a function from labeled training data. The training data consist of a set of training examples, each example is a pair consisting of an input object and a desired output value. A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalise from the training data to unseen situations in a "reasonable" way.

To solve a given problem of supervised learning, one must perform the following tasks:

* First the user should decide what type of data is to be used for the training set. If we are analyzing a person’s handwriting would we use a single character, a word or a full sentence of the handwriting. 
* Next, we need to gather a training set. The set needs to be representative of the real-world use of the function. Therefore, the inputs and outputs must be gathered either from human experts or from exact measurements.
* The correctness of the function will heavily depend on how the input object is represented. Usually, the input object is transformed into a feature vector, which contains several features that represent the object.
* We must also determine the structure of the learning algorithm. For example, the software engineer may decide to use decision trees to structure his algorithm. (see left)
* Finally, we must gauge the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set.

## Computational Intelligence

Computational intelligence refers to the ability of a computer to learn a specific task from data or experimental observation. It is used to address real-world problems to which traditional models can be useless. Traditional models can be useless when:

* The process is too complex for mathematical reasoning
* The process may contain some uncertainties
* The process may be completely random in distribution

As we know, many real-life problems cannot be converted into binary language for computers to process it. This is where Computational Intelligence can give us a solution to such problems.

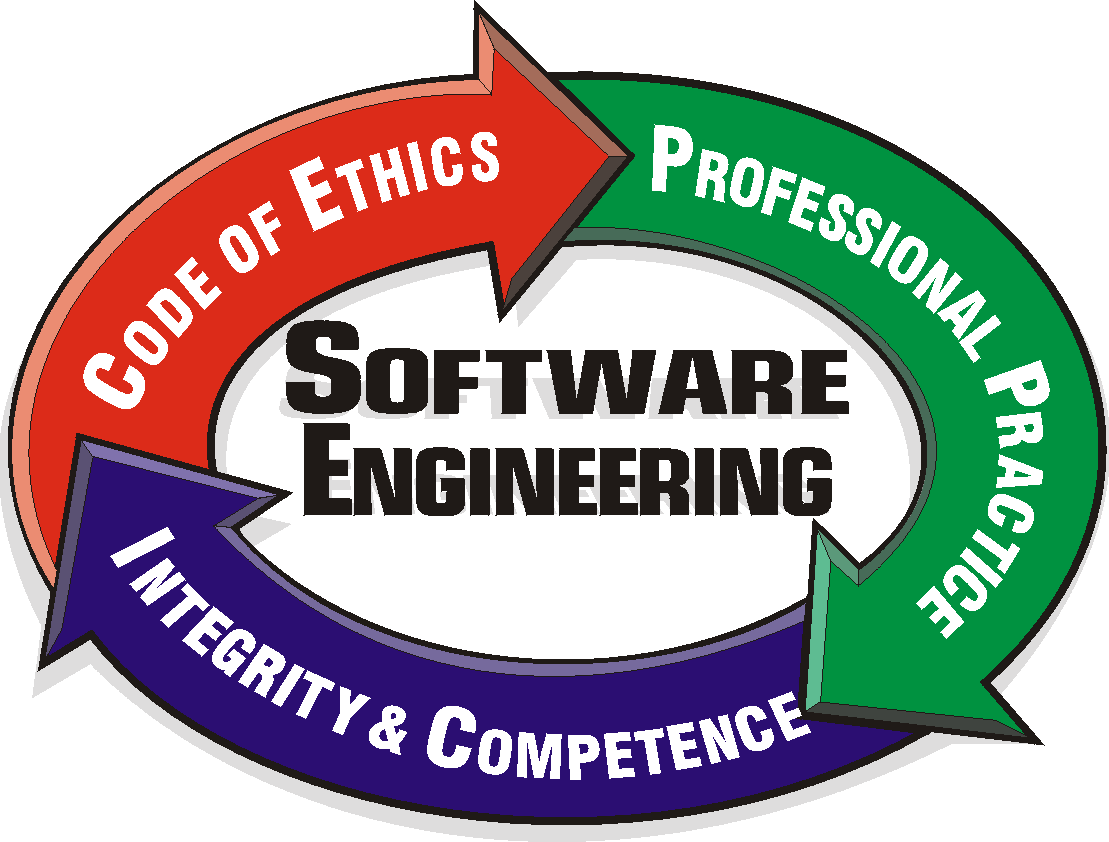
## 

# Ethics

## Ethics in Software Engineering

Ethics in the broadest sense refers to the concern that humans have always had for

figuring out how best to live. The philosopher Socrates is quoted as saying in 399 B.C.,

“the most important thing is not life, but the good life.” Engineering ethics is a well discussed topic and has been fully adopted for years. Most engineering ethics textbooks focus on the issues of mechanical or civil engineers. The classic case studies are that of the collapsing bridge or the malfunctional car with faulty breaks. One my ask how does this apply to software engineering? Well in the age we will live in today how many cars are built without complex software running them? How many bridges are built without using sophisticated computer programs to calculate the expected load, material strength and design resilience? How much of the flying of an airplane is done automatically by advanced software? A failure of these critical systems could be catastrophic and could cause just as many deaths as a faulty brake pedal or a loose bolt in an airplane. For this reason it is essential we study in detail the ethics of software engineering. 

## How can Software Engineers Harm the public?

As we stated above software engineers can be responsible for some catastrophic events that could cause great harm to the public. If such events directly or indirectly result from a software engineer’s choices to ignore their responsibilities, then these harms are the consequences of unethical professional behavior.

However, there are also less extreme ways software engineers can negatively affect the public that we must examine.

Example 1:

*Mary lives an incredibly hectic and stressful life, who needs to organize the free time she has more efficiently. She has just downloaded a new app called OrganiseMe onto her iPhone; this app merges information from Mary’s to-do list, information on her purchasing habits from retail stores she shops at, and GPS software to produce the most efficient map and directions for running errands on her days off. This collected data is not stored on Mary’s phone, but on a separate server that the app links to when it needs to create a shopping map. The app encourages users to log in via Facebook, as the developers have made a deal with Facebook to sell this data to third-party advertisers, for the purpose of targeting Facebook ads to Karen and her friends.*

Although nobody was seriously injured or killed the example above brings up some strong ethical concerns relating to software engineering. The thing that concerns me greatly about this example is that the Mary had no idea what was being done to her. This is where the ethical implications of software engineering can be even worse than the ethical implications of other engineering practices.

Worryingly this above example is commonly replicated in the real world. According to the New York Times in July of 2017 “The Chinese police said this week that they had arrested 22 people suspected of selling the personal data of an unspecified number of Apple customers. The police, in Cangnan County in the eastern province of Zhejiang, said the thieves had reaped 50 million renminbi, or about $7.3 million, over an unspecified period.”

## How can Software Engineers Bring Good to the Public?

When we talk about ethics must not only focus on avoiding harms, as I have discussed in detail above. We must also discuss the idea that software engineers must bring good to the world. Many of the comforts people enjoy today are down to complex software engineering projects. Whether it be the all-capable smart phones we rely on today, the ease of contactless credit card payments or the future of mini robots in your body performing surgeries nobody would argue that software engineering has changed the world we live in.

Software Engineers must continue to strive to innovate in order to continue to bring good to the public. In this effort to push the boundaries of technology our society will see many benefits.

## Beneficial or Detrimental?

In my opinion the monitoring of employees work is definitely beneficial to the area of software engineering. With continued monitoring I believe it will motivate software engineers to work honestly and to the best of their ability. A lack of monitoring can be detrimental to software engineering as we saw above in the Apple example in China. Although some argue that heavy monitoring may discourage or annoy software engineers and stop them working to the best of their ability. I believe the positives definitely outweigh the negatives.

# Bibliography

[**https://www.techopedia.com/definition/27913/technical-debt**](https://www.techopedia.com/definition/27913/technical-debt)

[**https://www.techopedia.com/definition/29893/analytics-as-a-service-aaas**](https://www.techopedia.com/definition/29893/analytics-as-a-service-aaas)

[**https://www.nytimes.com/2017/06/09/business/china-apple-personal-data-sold.html**](https://www.nytimes.com/2017/06/09/business/china-apple-personal-data-sold.html)

[**https://www.scu.edu/media/ethics-center/technology-ethics/Students.pdf**](https://www.scu.edu/media/ethics-center/technology-ethics/Students.pdf)