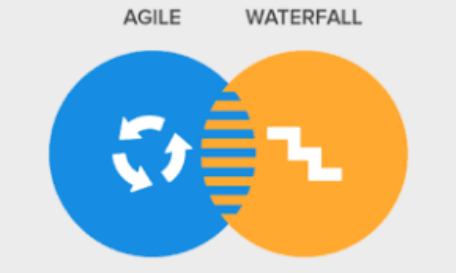
To deliver a report that considers the ways in which the software engineering process can be measured and assessed in terms of measurable data, an overview of the computational platforms available to perform this work, the algorithmic approaches available, and the ethics concerns surrounding this kind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

Measuring software engineering is essential in any organisation. Effective software engineering helps you identify what factors help and hinder a team's productivity[i]. The software engineering process has undergone many changes and there are a number of these that may influence the data to be used for measurement and collection. Some of these changes are as follows;

Waterfall to Agile

Traditionally, the software engineering process involved the Waterfall Model. This involves the following phases that are followed in sequential order.

· Requirements Analysis
· System design
· Implementation
· Testing/Verification
· Deployment
· Maintenance[ii]



While Waterfall is still popular in some organisations, Agile models such as Scrum, Kanban or Extreme Programming have replaced Waterfall. The main difference between Waterfall and Agile is that Agile is an iterative approach to writing software. Waterfall takes a predictive approach, and Agile takes an adaptive approach[iii].

The move from Waterfall to Agile affects how software engineering is measured. With Agile, there is a lot more interaction between software engineering teams and a lot more data 'footprints' where developers leave data about the work that they are doing i.e. a developer will commit code into a repository many more times in Agile than in Waterfall. This makes sense if you consider that Agile projects involve substantially more iterative albeit informal 'phases', so there is more data generated.

Programming Languages and Paradigms

Traditionally software engineering used programming languages such as Cobol to create applications. Each line of code was low level meaning that there was a huge amount of lines of code required to create that application. More modern programming languages like Python, Java or C# use libraries that contain predeveloped and pre-tested code meaning that one line of a developer's code could be multiple lines of hidden code. This means that we should not consider lines of code as being an important data point that needs to be captured[iv].

Software Engineering Tools

There is an increasingly large number of tools available to help run software engineering projects such as Bitbucket, Jira, GitHub or Microsoft DevOps. All these tools are designed to capture all the data footprints generated by developers assuming they are used correctly!

The Cloud 

The cloud presents lots of opportunities to improve Software Engineering, but in terms of measuring software engineering it provides a cheap and easy way of both extracting and storing the data from the tools mentioned above.

What data can we Collect and Measure?

If we are to consider what data can we collect, data can be collected at multiple steps of the software engineering process.

Data can be collected not just from the software engineers who are writing code, but from everyone in the software engineering process. Agile software engineering teams usually involve members from multiple disciplines so that they can perform all the duties required to get an iteration finished[v]. Some of these roles include, business analysis, development, testing, project management and even the end users of the software engineering product[vi]. Modern software engineering tools like GitHub are excellent in supporting all these various tasks that these roles perform[vii].

On the business end, analysts write requirements in the forms of user stories or use cases. These user stories are 'committed' when they get a status of complete. Usually by that stage there has been several changes, comments by other team members etc. All these actions create a tracking history associated with all the contributors but especially the business analyst.

Developers who write the code will usually commit their code when they have completed their individual tasks. However, once that commit is merged with other commits to build systems bugs may appear. In addition, if there are manual tests or

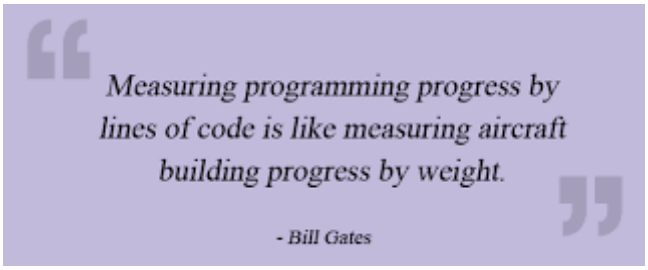
testers, they can assign bugs to the developers so they can fix the e code. All these interactions create a tracking history associated with all the contributors but especially the developer.

Developers use open source tools like GitHub to 'engage' with the software engineering world. It is important to recognise that this is much less important than writing some code that becomes part of a computer system.

Testers have their work 'committed' when their work gets a status of complete. Again, all these interactions create a history associated with all the contributors but especially the tester.

The end goal of software engineering is of course getting code into production. Once code is released, it is rare for live issues not to cause bugs[viii]. These bugs will normally be assigned to team members once again meaning more data is created that can be collected and measured[ix].

We are collecting a lot of data across the entire Software Engineering cycle. We could and should measure all of it. Ultimately all this data is captured at source and is available for further analysis.



Generally, we want to measure software engineering to improve performance. Deciding which data will we useful to do this is difficult. Measuring by lines of code, for example, can be misleading. Shorter code is usually more efficient as there is less chance to introduce bugs. There are many software metrics to track but ultimately, we only want to track efficient code. An alternative to lines of code produced might be lines of code produced that did not have to be altered or deleted[x]. It is important to choose metrics that are strongly correlated to the software goals. These metrics will differ from company to company, and from project to project, but ultimately should be relevant to how well the software user's requirements are met[xi].

How Can Software Engineering Be Measured?

It is vital to ensure that the tools being used to support the software engineers also cover the features to record that data. Such tools include Bitbucket, GitHub, Jira and Microsoft DevOps. The data in those tools should allow the data to be accessed (though APIs).

It is important that everyone on the team use the tools correctly. This is not down to the individual themselves, the tools themselves can be configured to do this to ensure that each developer does the likes of a commit when appropriate or a tester record a bug and assign it to the developer at the right time and place.

What platforms can be used to gather and process data?

Frameworks

A simple framework for data collection would be as follows;

- Identify the data sources. This would be GitHub or one of the other tools that software engineering teams are using.
- Identify what the data is that we require and confirm its availability. Data may have to be combined from different sources.
- Do we have all the data? If not, do we have to create our own supplementary data? We may have to identify a different source of data and in this case, its data that we create ourselves.
- The data may have to be transformed to a form we can use.
- Fixing the data: The data may have to be cleaned of outliers or made sure that it is reliable. We may have to make assumptions to ensure the data is complete. We may either must make assumptions to complete the data set or drop incomplete data sets.

Tools & Technologies

To implement frameworks, the following components can be considered;

APIs 

An API is a service that most modern tools and systems use to allow people to consume data from those tools and systems[xii].

Let's assume we're to using a tool to allow software engineers to capture all their footprints. Once that data is captured, that data must be available for consumptions using APIs. I've mentioned several tools above and all of those provide APIs that will allow us to get access to the data points that reflect the footprints.

ETL 

We can use tools categorised as ETL to Extract, Translate and Load the data that we collect from the APIs. ETL Tools are widely used technologies to allow us to read data from somewhere and load it into somewhere else. The Translate option allows us to cleanse the data, change the structure of the data or provide supplementary data that we'll need[xiii].

Database 

Data can be stored in databases. Two types of databases are relational and non-relational/NoSQL. Relational databases need to have strongly defined relationships between our data points. NoSQL databases allow us to store unstructured data[xiv].

The Cloud 

The cloud is the key platform that enables this entire process. All the services and tools that we need are easily available cheaply and at scale if we need it.

Personal software process (PSP)

A PSP involves gathering information about a developer's own work so that they assess their own ability.[xv] The developer can analyse their own strengths and weaknesses and adjust their software methods accordingly[xvi].

Some specific examples of platforms that gather and process data

Pluralsight Flow

PluralSight Flow is an organisational tool that analyses data from repositories and produces reports. Visualisations of work patterns are created allowing teams see what impacts productivity.[xvii]

CodeClimate

Code Climate is a platform that ensures code written is of good quality. They also provide a platform 'Velocity' that analyses data from GitHub repositories. It looks at similarities between pull requests that are rejected. This helps evaluate what coding techniques are producing more bottlenecks than others[xviii].

Mylyn (Previously Mylar)

Mylyn is a framework that collects and reports on trace information about a user's activity in Eclipse[xix]. Eclipse is generally used by Java Developers. Such a framework enables us to see how developers are using tools and whether they are using add-ons to their full advantage. One study by Murphy, Kersten, and Findlater[xx] looked at how different types of developers used different feature of Eclipse, such as the debugger, the variables view, the different commands they used etc. Such measurements indicate how tools can be improved to better accommodate the developer. Mylyn uses this data to filter your view of Eclipse so that it is more task focused[xxi]. The developer can be more efficient with their time.

What Algorithms Can we use? 

Once data is collected, it needs to be processed to extract meaning. An expert system is a computer system that can mimic the decision-meaning ability of a human expert[xxii]. It is purely mathematical and there is no human understanding involved. A software engineering measurement expert system tool (SEMEST) can be used to aid software engineering measurement[xxiii].

Multivariate Analysis

Multivariate analysis is used to study more complex sets of data. Multivariate analysis is extremely useful in the measurement of software engineering as we often record data from multiple sources, and it is vital to make sense of it all. The statistical program R is extremely effective as it provides multiple methods to perform statistical analysis on multivariate data.

Principal component analysis

One method of Multivariate Analysis is Principal component analysis (PCA). PCA is a form of dimension reduction that re-expresses the internal form of data in the form of linear combinations[xxiv]. The main goal is to reduce the number of variables in our data, while still preserving adequate information[xxv]. Regarding software engineering, if we were to measure multiple attributes of a team member to attempt to measure performance, we can apply PCA to reduce the dimensions of the attributes to those which have maximum correlation to their performance[xxvi]. We can also reduce the number of dimensions in order to pass the data into another algorithm.

Time Series Analysis and Forecasting

If we want to measure software engineering to make predictions about future data, Time Series Analysis can be used to;

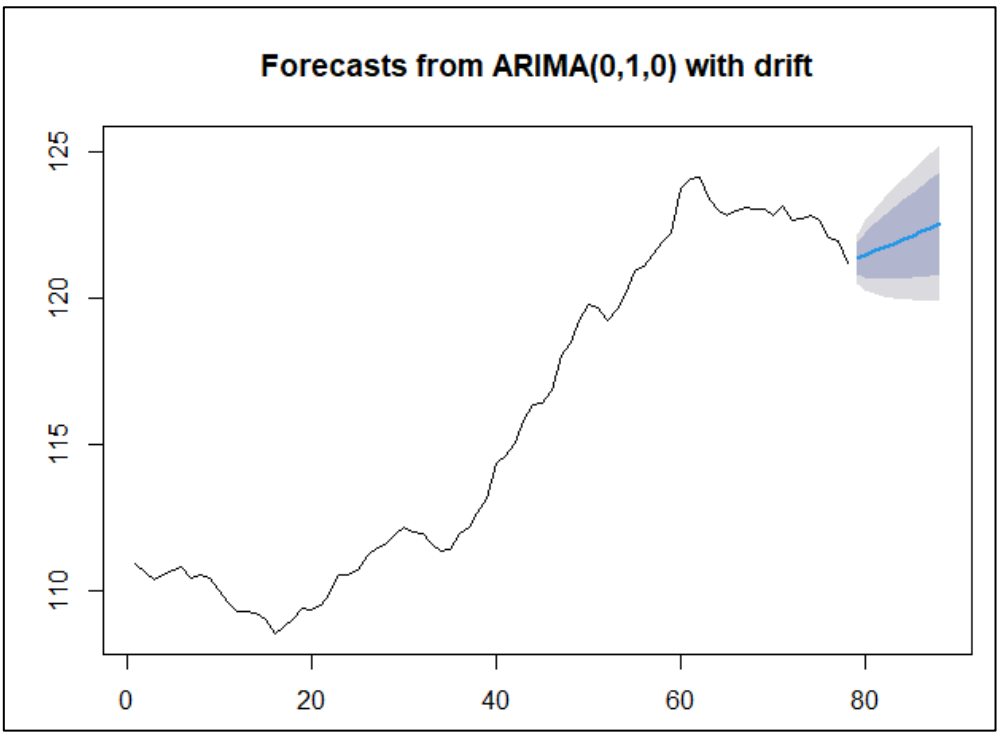
> <u>Understand the structure of observed data</u>
>
> Does our data contain elements of trend, a cycle, seasonality or random noise? For example, lines of code written by developers will follow a downward trend as developers will write more at a beginning of a project then lessen off as they only adjust existing code. The data may follow a daily cycle, writing more directly after a 'Scrum' meeting or in the morning when they are less tired. There may also be a weekly cycle – developers may be more productive towards the start of the week.

Fit a model to our data

Time Series analysis then allows us to fit an appropriate model to our data which can then be used to forecast future values. There are various models available such as HoltWinters single and double exponential smoothing, and ARIMA models. R provides packages that aid time series analysis and enable us to make forecasts.

Using time series, it would be possible to analyse and forecast when a developer will be more productive and try to cater towards their needs during that time, so that they have the largest impact.

The below graph gives an example of a plot generated using the R on some sample data. The grey areas signify different confidence intervals.



Using machine learning methods trained on past projects, we can try assessing how well current projects are going. If we are underperforming, we can then try identifying problems in the software engineering process.

The Ethical Concern

When evaluating the software engineering process, we are measuring the performance of the software engineers.

The purpose of measuring software engineering is analysis to aid decision-making. It is crucial that if patterns are being identified by computers, that people understand the process algorithms have gone through to identify patterns, especially if the pattern arrive at an important decision, the decision to sack someone for example![xxvii] If we use machine learning techniques leading to artificial intelligence or neural networks, can we be ever sure of the integrity of the results they produce? Such technology can be used for trend analysis but in my opinion, they can't be used for making decisions that may affect individuals. In this scenario, I'd have to suggest that they are unethical.

If we just measure software engineering in quantitative terms such as number of commits or lines of code generated are we ignoring equally important things such as how likeable a team member is or how much they contribute to morale? The influence of human interactions has been measured before. For example, some studies required people to wear sociometric badges at work to collect individual behavioural data[xxviii]. The task then is to use this known data to boost performance. An episode of the popular TV show 'Black Mirror' – 'Nosedive' has a plotline reminiscent of this. The society in the show used a technology where everybody rated their interactions with others on a scale of one to five stars, which affected your social status, job and overall, how you were treated. The question then is whether it is ethical to measure what essentially is one's personality. Also, is there really ever going to be an accurate way to do this?

[xxix]

Companies that use social technologies experience measurable business benefit[xxx] and there is no doubt that the measurement of software engineering can be extremely useful. One ever-present risk is the breach of confidentiality. When evaluating the software engineering process, the performance of the software engineers will inevitably be evaluated also. Any assessment of a software engineer based on measuring performance must adhere to the GDPR personal data rules. It's going to be very difficult explain to a software engineering team to get them to understand what their data is going to be used for. Data can be used to provide analysis on, say, how successful a software engineering project is. But if the datasets are there and we can use that to pinpoint how successful individuals are, I think this becomes unethical.

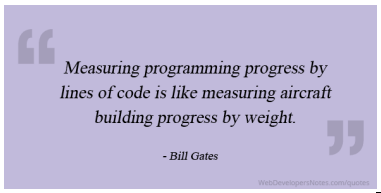
Most of these ethical concerns ask the same question – Just because we can do something, should we?

Software engineering is high risk because it is often the case you are developing new systems, with little or no existing guides or outlines. Measuring software engineering can enable us to perform analysis and reduce that uncertainty leading to better decisions[xxxi]. Software engineering is difficult to assess, but much needed to increase alignment between business and IT[xxxii].

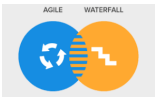
Image Links

 https://iconscout.com/icon/algorithm-2432362

 https://www.google.com/url?sa=i&url=http%3A%2F%2Fwww.hadoopsphere.com%2F2013%2F06%2Fbeyond-nsa-intelligence-community-has.html&psig=AOvVaw2SNvy8BEwYibLT-v55Dd02&ust=1606566444878000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCMCfk8rcou0CFQAAAAAdAAAAABAE



https://www.webdevelopersnotes.com/dont-measure-programming-progress-by-lines-of-code



https://www.bounteous.com/insights/2018/07/20/agile-vs-waterfall-which-best-your-team/

 https://www.informationsecuritybuzz.com/news/5-reasons-not-move-cloud/

https://icon-icons.com/icon/api/129131

 https://thenounproject.com/term/etl/2068754/

 https://icons8.com/icon/1476/database

References

i https://medium.com/@infopulseglobal_9037/top-10-software-development-metrics-to-measure-productivity-bcc9051c4615

ii https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm

iii https://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm

iv https://www.techrepublic.com/article/is-low-codeno-code-the-future-of-application-development/

v Multidisciplinary Teamwork in Agile and Scrum (methodsandtools.com)

vi https://www.methodsandtools.com/archive/archive.php?id=17

vii https://blog.zenhub.com/how-to-use-github-agile-project-management/

viii https://learn.marsdd.com/article/is-the-product-ready-for-release/

ix https://www.linkedin.com/pulse/application-support-agile-world-najaf-gillani

x https://stackify.com/track-software-metrics/#:~:text=For%20example%2C%20lines%20of%20code,%E2%80%9Cdead%20code%E2%80%9D%20or%20comments.

xi https://stackify.com/track-software-metrics/#:~:text=For%20example%2C%20lines%20of%20code,%E2%80%9Cdead%20code%E2%80%9D%20or%20comments.

xii https://www.mulesoft.com/resources/api/what-is-an-api

xiii https://en.wikipedia.org/wiki/Extract,_transform,_load

xiv . https://en.wikipedia.org/wiki/Database_schema https://www.mongodb.com/nosql-explained

xv Professor Stephen Barrett in Face to face live session for CS3012 - Software Engineering TCD

xvi Microsoft PowerPoint - PSP Tutorial part 1.ppt (sigada.org)

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

xviii https://codeclimate.com/velocity/understand-diagnose/#overview

xix G. C. Murphy, M. Kersten, and L. Findlater, “How are Java software developers using the Elipse IDE?” IEEE Software, vol. 23, no. 4, pp. 76–83, Jul. 2006.

xx xx G. C. Murphy, M. Kersten, and L. Findlater, “How are Java software developers using the Elipse IDE?” IEEE Software, vol. 23, no. 4, pp. 76–83, Jul. 2006.

xxi Eclipse Mylar Introduces Major New Innovation for Developer Productivity | The Eclipse Foundation

xxii Professor Stephen Barrett in Face to face live session for CS3012 - Software Engineering TCD

xxiii (PDF) A Web-based software engineering measurement expert system (researchgate.net)

xxiv Arthur White – Slides In STU33011 Multivariate Analysis TCD
xxv Arthur White – Slides In STU33011 Multivariate Analysis TCD
xxvi https://ieeexplore.ieee.org/document/7228109

xxvii Professor Stephen Barrett in Face to face live session for CS3012 - Software Engineering TCD

xxviii

https://dspace.mit.edu/bitstream/handle/1721.1/92443/Pentland_Modeling%20dynamical.pdf?sequence=1&isAllowed=y

xxix https://collaborativedialogues.wordpress.com/2017/01/05/nosedive-black-mirror-technologys-influence-on-life-today/

xxx http://www.nextlearning.nl/wp-content/uploads/sites/11/2015/02/McKinsey-on-Impact-social-technologies.pdf

xxxi https://hackernoon.com/3-reasons-to-use-forecasts-in-software-development-b09a186e044d

xxxii https://repository.tudelft.nl/islandora/object/uuid%3Ae91b25d0-7639-4591-a067-dc06bcbcf39f