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

Paul O’Callaghan
18324372

<u>Consider 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.</u>

Software engineering in terms of measurable data:

Software engineering encompasses how we use and utilize all forms of data. Whether it involves using measurable data to quantify a problem in order to view many aspects of this problem or using qualitative data to determine a particular frequency of characteristics in order to allow larger sets of data to be observed, data is a fundamental part of software engineering.

Quantitative data is an area I find particularly fascinating, it accounts for data that has a unique numerical value associated with it, such as the weight or height of a particular individual. The term software metrics is often used to describe the way in which we measure quantifiable data.

The ultimate goal of software metrics is to analyze the quality of products with the aim of providing critical feedback, improving the quality of the product, and predicting its quality once the software development process is complete.

Software metrics have provided an abundance of benefits to software engineering. Managers now utilize software metrics to prioritize and track any issues allowing for better efficiency between teams. Not only does this result in a huge spike in productivity but the software engineering process becomes cheaper as software engineers can detect issues at an earlier stage, leading to an increase in return on investment (ROI).

Software metrics provides a platform for higher quality communication between teams. This assists in monitoring and detecting issues at the earliest stage allowing for a better workflow. It also assists managers in assessing goals and objectives throughout projects allowing them to identify areas of improvement, manage workers' work loads and reduce overtime.

Although software metrics increases ROI, identifies areas of improvement and manages workloads, it coherently lacks clarity due to multiple terms and characteristics associated with it. An example of this is Line of Code (LOC). LOC is a commonly used practice for measuring software development. It calculates the remaining work left to be completed by engineers in order to complete the project.

However, the process of counting lines for code can be completed in a variety of ways. Different methods yield contradicting results leading to a lack of clarity. This is a direct example as to why a standard for software measurement should be defined at the beginning of the project and consistently be carried out.

For example, LOC can be completed in two ways.

1. Physical LOC measures lines from the source code files. However this includes both empty lines and commented lines leading to an inaccurate total.
2. Logical LOC counts the number of lines with an element specific to each language ie. Java/C/C++ measures the number of semicolons to estimate the number of statements. In my opinion, Logical LOC is an optimal approach and yields a more realistic outcome.

Issues also exist surrounding the usage of software metrics. Software engineers can actively solve less complex problems to maintain a high LOC and low error count. This ‘trick to the system’ deceives managers and gives a perception that the group is operating at high productivity and can result in delays to project completion along with unhappy customers.

An overview of computational platforms available

There are many computational programs that assist the measurement of software engineering. Such programs include the Personal Software Process (PSP) whose structured process is designed to educate engineers and improve performance; Hackystat which provides a framework for collection, analysis, visualization, interpretation and annotation of software development process; and GitPrime, a cloud based service that analyses information from version control systems used by companies.

Below I have analysed Hackystat and GitPrime along with a detailed description of the Personal software Process.

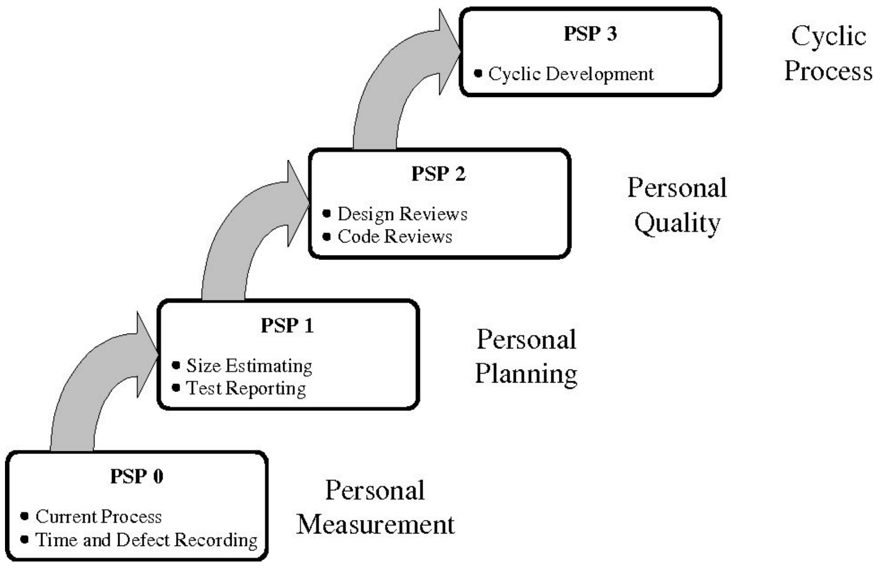
Personal software Process (PSP)

The PSP was originally developed by Watts Humphrey who. Humphrey was an american pioneer in software engineering. Many consider him the “father of software quality”. The Personal Software process is a measured software process designed for the purpose of assisting software engineers in their work.

PSP enhances improvement by extending process management and control to the software engineer. With PSP, software engineers develop software using a disciplined and structured approach. They follow a defined process, plan, measure, and track their work, manage product quality, and apply quantitative feedback to improve their personal work processes leading to better estimating planning and tracking.

PSP requires participation by all management levels for optimal performance. Management staff must be involved first and then engineers can be trained on a project-to-project basis.

Below is a diagram illustrating the PSP



Quality Management

Poor quality management has many negative effects on an organization. It increases software development costs, and makes schedules even harder to predict leading to a lack of clarity amongst engineers. Many problems are a direct result of the reliance on unit testing as a measure for the quality of software produced. However rectifying defects in tests is costly, ineffective, and unpredictable.

The optimal method to manage software quality is to locate and remove defects in a program. PSP assists engineers in locating defects before compilation, unit testing or system testing. As a result PSP significantly reduces the number of defects in a system and contributes to a higher accuracy in schedule predictability.

Cycle Time

PSP decreases the cycle time for a project. Cycle time can be greatly reduced by better quality planning and the elimination of rework. Teams producing accurate plans allows for tighter scheduling coupled with greater interaction among meetings and planned activities.
Locating and removing defects at an early stage is crucial. It allows for better quality and further increases the accuracy of planning by limiting variation.

The PSP educates engineers in gathering and processing data required to reduce the life cycle. This data also eliminates rework, assists them in building accurate plans and reduces system testing by a maximum factor of 5 times.

Cost Management

Cost and schedule problems originate when projects make commitments that are based on inadequate estimates of size and development resources. PSP addresses this problem by showing engineers how to make better size and resource estimates using statistical techniques and historical data.

An initial bias existed towards underestimating has since evolved to a more balanced mix of overestimates and underestimates. A more balanced estimation error means that the errors tend to cancel, rather than compound, when multiple estimates are combined.

Hackystat

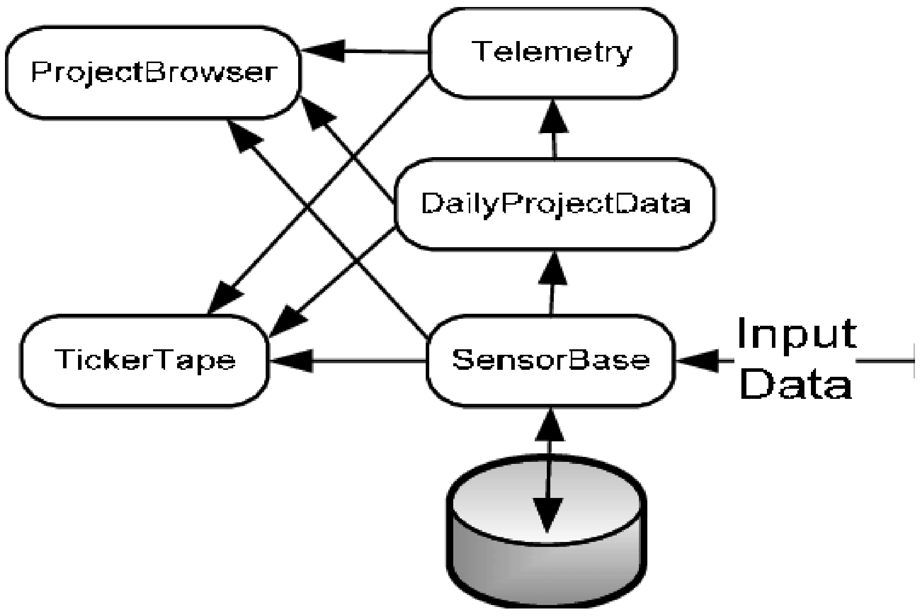
Hackystat is an open source framework for collection, analysis, visualization, interpretation, annotation, and dissemination of software development process and product data. The platform supports three different areas of software engineering.

1. Researchers - It can be used to support metrics validation, software engineering experimentation as well as long range research initiatives such as collective intelligence.

2. Educators - It is commonly used in software engineering courses at all levels to introduce students to software measurement.
3. Practitioners - It can be used as infrastructure to support professional development by facilitating the collection and analysis of data. Hackystat is also useful for quality assurance, project management and resource allocation.

Hackystat aims to facilitate “collective intelligence” in software engineering by enabling collection, annotation, and diffusion of information and its subsequent analysis and abstraction into useful insight and knowledge. Their services also co-exist and complement other components of the cloud internet information system.

<u>Below is a diagram illustrating Hackystat</u>



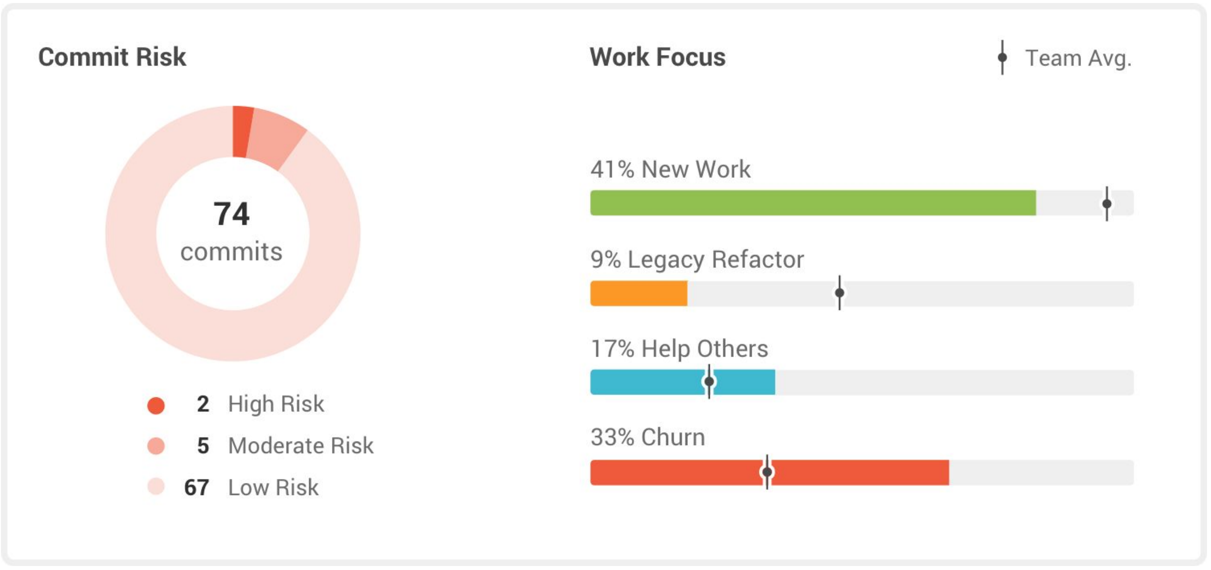
GitPrime

GitPrime gathers and utilizes data from GitHub, GitLab, BitBucket - or any Git based code repository - to assist engineers in moving faster, optimize work patterns, and advocate for engineering with concrete data.

GitPrime analyses a wide range of aspects of code.

GitPrime also views a project's code commits, ticket activity, and PRs in a single dashboard. This allows managers to analyse engineers' work in a time efficient manner. It also identifies project bottlenecks at an early stage. A project bottleneck is a process in a chain of processes that reduces the capabilities of the total chain due to its own limited capabilities. For example, an inefficient machine will have a long queue.

<u>Below is an example illustrating the information analysed by GitPrime</u>



Algorithmic approaches

Artificial intelligence and machine learning platforms have the ability to process huge data sets at any given time. It has provided many areas with continuous growth since its inception, such as chatbots to interact with users; platforms to assist workplace communication resulting in optimal efficiency and human resource management.

Machine learning is comprised of two main areas:

1. Supervised learning
2. Unsupervised learning
3. Statistical analysis

Supervised learning

Supervised learning involves training the machine using data that is appropriately labelled. This data is already tagged with the correct answer which allows the machine to make comparisons which occurs in the presence of a supervisor.

A supervised learning algorithm learns from labeled training data. This assists in predicting outcomes for unforeseen data. Successfully building, scaling, and deploying accurate supervised machine learning data science models takes time and technical expertise from a team of highly skilled data scientists.

Supervised learning is very beneficial as it allows users to collect data or produce an output from the previous experience. This experience also assists in optimizing performance criteria and solves real world computational problems of various types.

Unsupervised Learning

Unsupervised learning is a machine learning technique, where you do not need to supervise the model. As an alternative approach, the machine is allowed to work independently to discover information freely. This approach is mainly associated with unlabelled data.

It allows users to solve problems of greater complexity in comparison to supervised learning. However, unsupervised learning can be more unpredictable compared with other natural learning deep learning and reinforcement learning methods. The unsupervised approach can detect an abundance of abstract patterns in data and can assist in locating features suitable for categorization.

Ethics Surrounding Software Engineering

Privacy Issues

Privacy, particularly software privacy, is an area of software engineering ethics that I find fascinating. Software engineering has seen the emergence of software privacy and privacy engineers due to recent developments in General Data Protection Regulations (GDPR). Privacy software is software built to protect the privacy of its users. The software typically works in conjunction with Internet usage to control or limit the amount of information made available to third parties.

Despite the emergence of privacy engineers, the number of data breaches continues to increase in both number and sophistication every year causing an increased financial burden on individuals and multinational companies. A recent study from the University of Maryland found that a successful hack is completed every 39 seconds due to our use of non secure usernames and passwords.

Currently, location services is one of the biggest ethics concerns surrounding software engineering for a variety of reasons. Today, many applications require a user's location for additional features such as Snapchat, Facebook and Instagram. Personally, my biggest concern regarding location services is the normalization which is taking place in society. People around the world accept and are even oblivious to the number of applications accessing our location. For example, the infamous SnapMaps gained much notoriety when Snapchat announced it’s latest feature release however today, the idea of many people having access to our location freely has become normalized and could present a great threat to society if this trend continues.

Data Analysis

A major ethical concern associated with software engineering is how data is analysed and interpreted. Today we place an increasingly large level of trust in machines and in their ability to interpret data. However, machine learning algorithms lack accuracy in particular with predictive analytics leading to inaccurate information. Common types of errors in machine learning algorithms include in-sample and out-of-sample errors.

References:

https://en.wikipedia.org/wiki/Source_lines_of_code#:~:text=Source%20lines%20of%20code%20(SLOC,of%20the%20program's%20source%20code.

https://resources.sei.cmu.edu/asset_files/TechnicalReport/1997_005_001_16565.pdf

https://hackystat.github.io/#:~:text=Hackystat%20is%20an%20open%20source,Researchers.

https://stackshare.io/stackups/github-vs-gitprime

https://en.wikipedia.org/wiki/Bottleneck_(production)

https://www.guru99.com/supervised-vs-unsupervised-learning.html

https://eng.umd.edu/news/story/study-hackers-attack-every-39-seconds

https://www.business.com/articles/how-to-maintain-data-privacy-during-software-development/

https://en.wikipedia.org/wiki/Privacy_engineering

https://www.innoarchitech.com/blog/machine-learning-an-in-depth-non-technical-guide-part-4#:~:text=There%20are%20multiple%20types%20of,used%20to%20build%20predictive%20models.