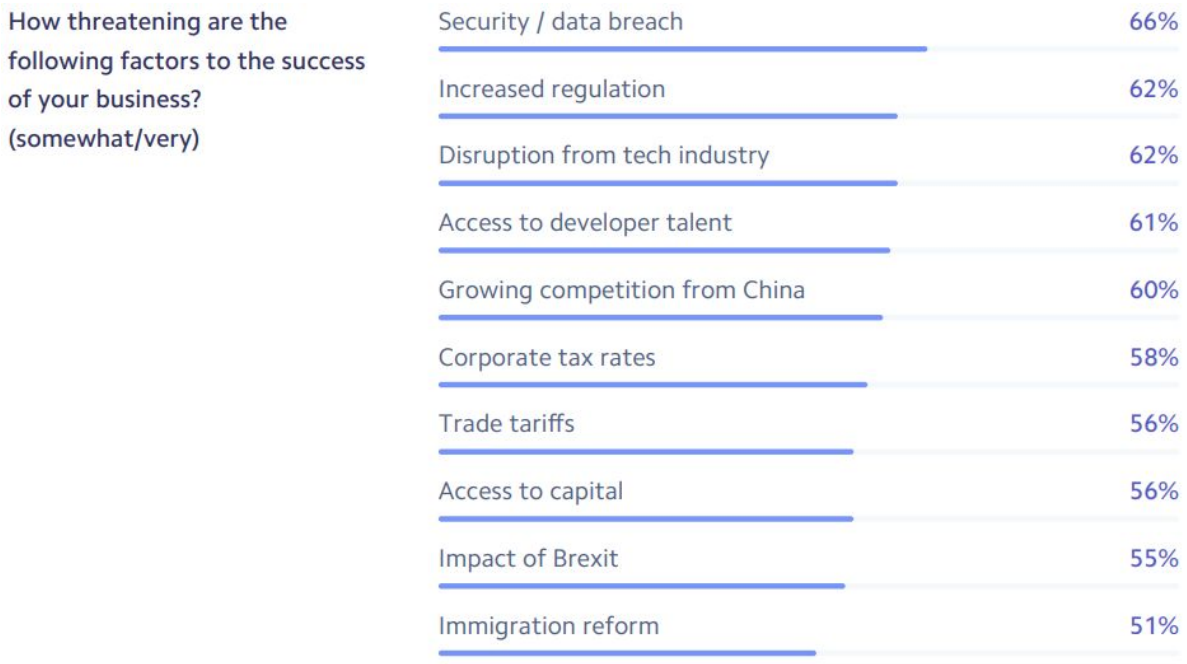
Report on the Measurement of Engineering

Overview

The purpose of this report is to consider the ways in which the software engineering process can be measured and assessed in terms of measurable data. In doing this, I will provide an overview of the computational platforms available to perform such work, and the algorithmic approaches available. We will also explore any ethical concerns surrounding this kind of analytics.

Section 1: Measurable Data

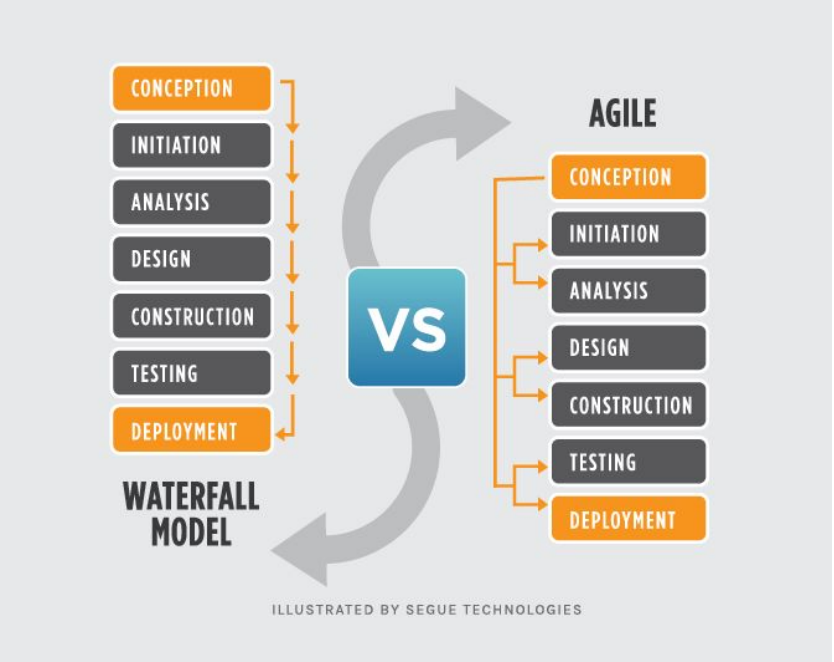
Software metrics is the term used to describe activities concerned with the measurement of software engineering. These metrics provide information to support managerial decision-making during software development and testing (Fenton & Neil). The importance of productivity in software engineering may prove to be more important now than ever before, with claims that senior executives now worry more about access to developers than they worry about access to capital, immigration concerns, and other challenges (Stripe).



The British quality standard for systems and software engineering, ISO/IEC 25010:2011, last reviewed in 2017, defines attributes of software products by which quality can be systematically described. The standards provide a product quality model which categorises system/software product quality properties into eight characteristics as follows, each composed of a set of related sub-characteristics: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability and portability.

The standards are clear when it comes to measuring the quality of the outputs of the software engineering process, but how can we measure the process itself? We can begin by defining what exactly is meant by the “software engineering process” to gain some insight into what we want to measure.

A software engineering process is the model chosen for managing the creation of software from initial customer inception to the release of the finished product. The chosen process usually involves techniques such as analysis, design, coding, testing and maintenance (Hawkey). There are a number of different models used in software engineering, the waterfall model or the agile methodology for example, which may be seen to provide metrics in itself.



(Source: Segue Technologies - https://www.sequetech.com/waterfall-vs-agile-methodology/)

Agile Metrics

Having spent 12 weeks working as an intern in an agile environment under a scrum framework, I think it is fair to say that a lot of emphasis was placed on agile development practices. How many of our assigned development tasks were we able to complete in each sprint? How quickly were we able to resolve any defects or concerns raised? Agile metrics include lead time, cycle time and velocity. Metrics like these are useful in planning and decision-making, but shouldn’t necessarily be used to measure success, value added, or software quality. For example, it would not be advisable to measure productivity using story points, which are used predominantly for self-reporting. Measures that can be overestimated or underestimated provide no true reflection of work completed, and would therefore, in my opinion, be a poor measure of productivity, from a managerial perspective.

Lines of Code

From a leadership stance, the lines of code (LOC) metric is still routinely used in industry to assess productivity of employees and provide crude cost/effort estimates. While this is an easy metric to compute and understand, it may not be the most effective measure. Used in isolation, simple metrics like size counts (e.g. LOC) and defects counts (e.g. defects per KLOC) have limitations and are commonly misapplied. Increasing diversity in programming languages also contributes to the problems of the LOC measure. Lines of code written in an assembly language will not be comparable in effort, functionality, or complexity to that of a high-level language (Fenton). Aside from this, quantity does not always equate to quality.

Code Churn

Code churn can be useful in assessing progress and understanding among software engineers. It measures the rate at which your code evolves i.e. lines added, modified or deleted from one version to another. While testing and reworking code are normal parts of the development process, deviations from an engineer's normal level can indicate problems. Higher code churn, for example, may indicate that employees are struggling with a particular task, that they are unclear on the goal of the software, or that requirements are changing frequently, which leads to a lot of wastage.

The following factors can cause unusually high levels of code churn (GitPrime):

1. Prototyping - When an engineer is coming to terms with a new or unfamiliar project, it is natural for them to test multiple solutions, causing increased code churn levels. The prototyping process is a common pattern, and engineers shouldn't be interrupted unless time expectations are exceeded.
2. Perfectionism - Perfectionism arises when an engineer is rewriting large portions of code in the later stages of the project without adding much functionality or value. This time could likely be spent more productively elsewhere.
3. Issues Concerning External Stakeholders/Unclear Requirements - Unclear or indecisive stakeholders can be very damaging to team morale, hindering progress and causing frustration. The problem faced by the team should be shrinking over time, as a solution is gradually built. A spike in code churn towards the end of a project could suggest that new requests have been made, changing the initial requirements. If specs are poorly written, engineers may work under certain assumptions which may later prove wrong, causing a need to rewrite code.
4. Difficult Problem - With more challenging problems, we can expect more exploration and reworking of code until a solution is reached. This is natural, but if the problem persists, it should not be left unchecked for too long.

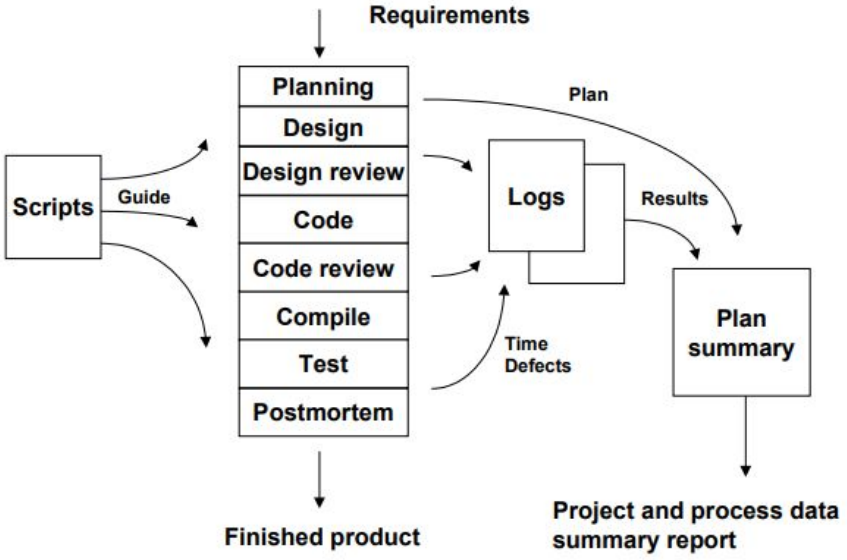
Section 2: Computational Platforms

Moving on from the actual metrics being taken, we will explore the ways in which large volumes of data can be collected and analysed. Increasingly, companies are using open source platforms for analysis, both of productivity and other forms of data. For the purpose of this report, we will first look at older platforms, working our way up to the computational platforms currently in use.

Personal Software Process

The Personal Software Process (PSP) is used in software engineering to manage quality, analyse the results of each job, and use these results to improve the process for the next project. It shows engineers how to plan and track their work, use a defined and measured process, establish measurable goals, and track performance against these goals (Humphrey).

The PSP is based on the concept that every engineer is different, and plans should be based on their own personal data. Engineers should feel personally responsible for the quality of their products, and should log each step of the process to track their individual productivity. The PSP process flow, as shown in Humphrey's report, is detailed below:



A report from the University of Hawaii states that most students using the PSP "were able to estimate both the size and time of their final project with 90% or better accuracy, and one student achieved 100% yield on one project (This means that the student eliminated all syntax and semantic errors from the system prior to the first compile of that system.) ." In spite of these results, however, none continued to use the PSP after the semester, likely due to the overhead involved in the manual collection and analysis required by the personal software process. (Johnson, Kou et al.)

Leap

The Leap toolkit was developed to lower these overheads in collection and analysis of individual software engineering metric data. This “second generation approach” still requires manual recording of effort, size and defect information, but significantly lowers the overhead associated with metric analysis, offering automated support. As with the PSP, this process was taught at the University of Hawaii, where a small number of students continued to use Leap after the end of the semester. Again, the requirement for manual data entry acted as a barrier to greater adoption of the technique, the continual context switching between doing work and recording work too intrusive for many users. (Johnson, Kou et al.)

Hackystat

Hackystat effectively eliminates the overhead of metrics collection, using client-side sensors that attach to development tools and unobtrusively collect effort, size, defect, and other metrics regarding the user’s development activities. Once the user has registered with a Hackystat server and installed the sensors, metrics are collected and sent to the server at regular intervals, where analysis programs are run over all the metrics for each developer. Hackystat is useful in tracking progress, with tools such as the “Daily Diary” which represents the developer’s activity at five minute intervals over the course of a day, allowing for the generation of other analyses such as developer effort on certain modules per day, change in module size per day, and unit test outcomes. Hackystat also provides the ability to define alerts, so the user doesn’t have to focus on the metrics until necessary. Of course, there are also concerns regarding the adoption of Hackystat, mainly surrounding privacy-related issues due to the continuous collection of data. (Johnson, Kou et al.)

Code Climate

Code Climate is popular among engineering teams, and consists of two main products. ‘Quality’, an automated code review tool, is intended to help developers improve the maintainability of their codebases. ‘Velocity’, the company’s flagship product, analyses data stored in GitHub repositories to create reports and provide prospective on how individuals or, more commonly, teams are working. (Code Climate)

GitPrime

GitPrime was designed to provide better metrics to improve the measurement and communication of productivity in software engineering. Rather than relying on subjective metrics like story points and tickets cleared, GitPrime mines data in Git to measure productivity and collaboration (GitPrime). The company even promises a guaranteed 20% increase in productivity (GitPrime for Executives).

Jira

Jira is a tool used primarily in agile teams. Jira Software allows users to plan, track, and release software, and provides visual data reports so teams can assess their performance and productivity. (Jira)

Conclusion

The evolution of the platforms used in the collection and analysis of productivity data is clear in this section, highlighting the move away from manual, personal data collection towards open-source platforms which automate the process

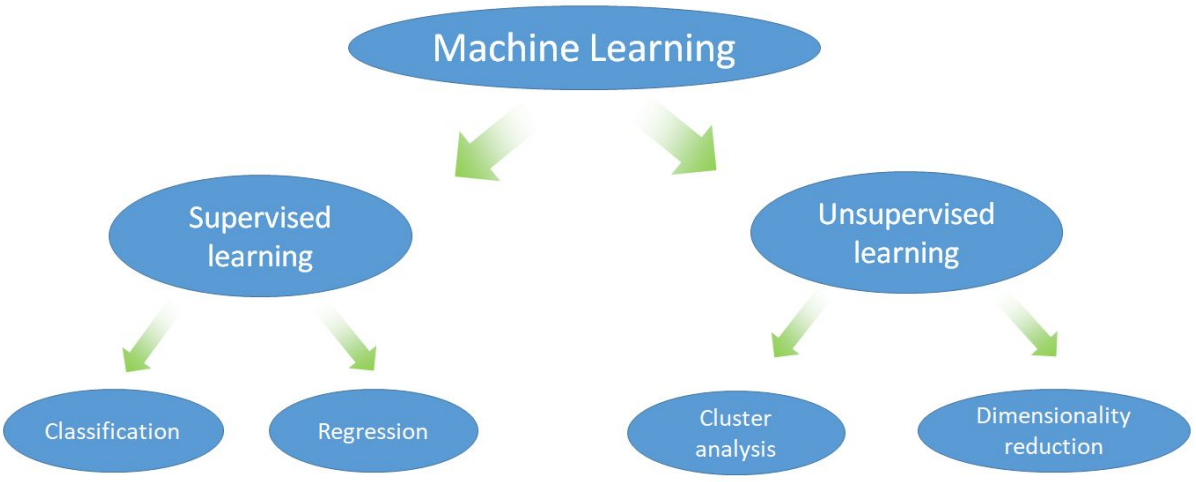
Section 3: Algorithmic Approaches

Having gained a better knowledge of the metrics and computational platforms used in measuring the software engineering process, we shall now explore the algorithmic approaches available.

There are, of course, various algorithmic approaches to analysing productivity data. However, with growing interest in artificial intelligence, it seems most appropriate to examine methods that fall within this area. Thus, for the purpose of this report, I will discuss machine learning approaches to data analysis.

"Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention."
(SAS)

Two of the most popular machine learning methods are supervised learning and unsupervised learning.



(Source: http://www.diegocalvo.es/en/machine-learning-supervised-unsupervised/)

Supervised Learning

Supervised learning algorithms use labeled examples, taking a set of inputs along with the corresponding correct outputs. By comparing actual output with correct outputs to find errors, the algorithm learns and modifies the model accordingly. Supervised learning is often used in applications where historical data predicts likely future events (SAS). Supervised learning uses classification and regression techniques such as linear discriminant analysis and the k-nearest neighbour method.

K-Nearest Neighbours Classification

The k-nearest neighbour classification method is non-parametric in that it makes no assumption on the spread of data within each class. This makes it a very simple method with fixed class assignment and no measurement of uncertainty concerning any particular assignment. After a positive integer value for k is specified, the class assignment of the k closest neighbours of each new point is considered. The new observation is then classified as belonging to the most prevalent classification of these entries.

Linear Discriminant Analysis

Discriminant analysis uses the class information to help reveal the structure of the data, allowing for the classification of future observations. Linear Discriminant Analysis (LDA) is a classification technique that assumes the use of a distribution over the data, enabling us to quantify our uncertainty over the structure of the data. LDA, unlike quadratic discriminant analysis, assumes groups have the same covariance matrix.

Unsupervised Learning

Unsupervised learning algorithms use data with no historical labels. The algorithm is not told the “right answer”, as with supervised learning, but must figure out what it is being shown and find some structure in the data (SAS). Unsupervised learning uses clustering and dimensionality reduction techniques such as hierarchical clustering and principal component analysis.

Hierarchical Clustering

The aim of cluster analysis is to establish whether there is a group structure in the data, and if so, to find the number of groups present and their particular structures. Hierarchical clustering starts by assigning each observation to a group on its own. The two closest groups are then combined into a single group, leaving one fewer group. This process is repeated until only one group remains.

Principal Component Analysis

Principal component analysis (PCA) is a dimension reduction technique that finds linear combinations of variables in the data which capture most of the data’s variation. Principal components are found by performing an eigen-decomposition of the data’s covariance matrix.

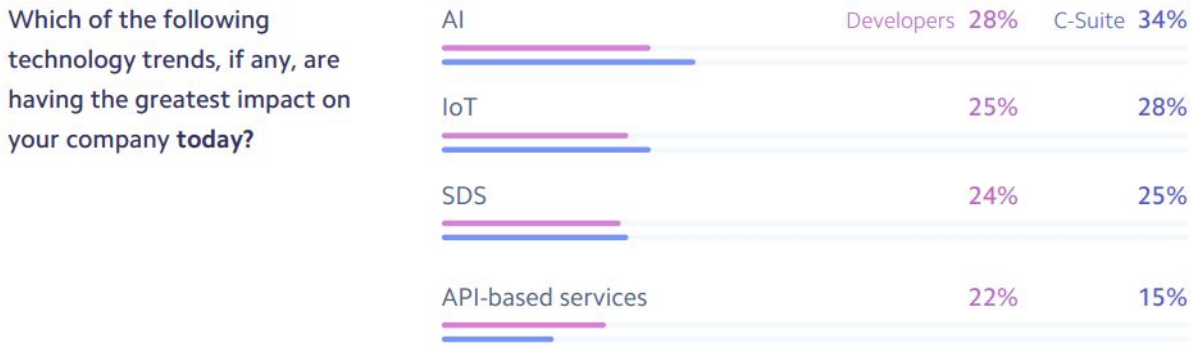
Conclusion

Much of the data analysis required in today's world can be carried out by artificial intelligence. Machine learning has grown in popularity due to growing volumes and varieties of data, cheaper and more powerful computational processing, and affordable data storage. Most industries working with large amounts of data have recognised the value of machine learning technology, and the competitive advantage to be won by using some of the algorithms outlined above. (SAS)

Section 4: Ethics

Now that we have explored the measurement of the software engineering process from a technical perspective, we must reposition ourselves and discuss the ethical side of things. Data collection and analysis is often a controversial subject, but how does computational intelligence impact the moral principles surrounding data collection and analysis?

In the aforementioned study conducted by Stripe, developers and C-level executives were quizzed on the technology trends most impacting their company. The top response from both developers and executives was artificial intelligence, a testament to the rising interest in AI in recent times and its potential to change the way we work in future.



In a discussion surrounding their book, 'The Future of the Professions', Richard and Daniel Sussman contemplate whether there may be entirely new ways of organising professional work as we move from a "print-based industrial society to a technology-based internet society" (Sussman). In an era when machines can out-perform human beings at most tasks, who should own and control online expertise, and what tasks should be reserved exclusively for people? Most people would be understandably nervous in handing over responsibility to machines when it comes to certain areas of our lives. Should it be left down to a machine, for example, to decide on the appropriate time to turn off a life support system? A thought from Richard Sussman on the subject of using artificial intelligence in decision-making really grabbed my attention:

“The question is not ‘how do you replicate a human being’s exercise of judgement?’, it’s ‘can we develop systems that can manage the uncertainty that we as human beings handle through human judgement?’.”

The implication here is that the only difference between machines and us as humans, is our ability to foresee and cope with uncertainty. Is it possible that such systems could be developed to eradicate this difference, and if so, where should moral limits lie?

On the one hand, technology and artificial intelligence are changing our understanding of employees and productivity for the better. In a Hitachi study, the effects of happiness on productivity were measured using wearable technology, which also uses artificial intelligence to generate key performance indicators. The research suggests that physical activity, happiness, and productivity are all correlated (Yano, Akitomi et al.).

IBM claim they can use AI to predict which workers are about to quit their job with roughly 95% accuracy. The company has performed a massive overhaul of its human resources department, believing it to be an area where “humans need AI to improve the work”, reducing its size by 30%. The artificial intelligence in use has so far saved IBM nearly $300 million in retention costs, and also identifies employee skills and areas for improvement so they can be directed towards opportunities they may have missed using traditional methods. (Rosenbaum)

The advantages of both these applications is clear, both for employees and management. But not all applications of AI in data collection and decision-making are viewed the same way.

Earlier this year it was revealed that Amazon used a computer system to “track and fire hundreds of fulfillment center employees for failing to meet productivity quotas” (Tangermann). Many ex-employees have come forward to discuss Amazon’s infamously inhumane working conditions, with protests arising over insufficient break time and constant monitoring and supervision by AI systems.

Another interesting situation arose in 2018, when Ibrahim Diallo was fired by his company’s artificial intelligence system due to an administrative error. Diallo arrived at work unaware of his termination, much like his bosses, who could not figure out why his contract had been terminated. The computer system revoked his access to the building, his computer and all company accounts, and “no human could do a thing about it” (Diallo). This story raises questions surrounding the power delegated to computational intelligence systems, and the ability (or inability, in this case) of humans to overwrite any decisions made by AI.

"*Automation can be an asset to a company, but there needs to be a way for humans to take over if the machine makes a mistake*"
(Diallo)

I believe that, in order to move forward with further implementation of AI in the workplace, more safeguards must be put in place to allow for human interference where these systems may make an incorrect decision.

Where data is concerned, the issue of privacy will always arise. The introduction of the General Data Protection Regulation (GDPR) by the European Union in 2018 has greatly strengthened the protections surrounding data collection. GDPR requires greater transparency from companies about the data they hold, and how they store and use it, providing some peace of mind to individuals.

We previously mentioned the privacy concerns surrounding the continuous collection of data in Hackystat. With the growth in prominence of big data collection and analysis, many people feel their rights have been violated at some point. With the Cambridge Analytica story breaking in 2018 and ongoing investigations into Facebook's breach of data law, one has to question whether individuals should be able to withhold more of their own data. However, in a world where people must offer up data in order to advance, both professionally and socially, I personally struggle to see how this is possible.

Summary

In conclusion, there are a number of different approaches to measuring the software engineering process, in terms of the metrics used, the computational platforms available, and the algorithmic approaches used. The rise of computational intelligence has and will continue to impact the way in which we collect and analyse data, and the way we work in general. As we delve deeper into using artificial intelligence in data analysis and decision-making, it is important to maintain ethical standards, which are both legally and morally upstanding.

Bibliography

Anaxi, “Software Engineering Metrics: The Advanced Guide” [online]
https://anaxi.com/software-engineering-metrics-an-advanced-guide/
[Accessed 27/10/2019]

British Standards Institution, BS ISO/IEC 25010:2011 [online]
https://pdfs.semanticscholar.org/57a5/b99eceff9da205e244337c9f4678b5b23d25.pdf
[Accessed 25/10/2019]

Code Climate. “Our Story” [online]
https://codeclimate.com/about/
[Accessed 2/11/2019]

Diallo. Diallo, I. (2018) “The Machine Fired Me” [online]
https://idiallo.com/blog/when-a-machine-fired-me
[Accessed 2/11/2019]

Fenton & Neil. Norman E. Fenton, Martin Neil, The Journal of Systems and Software 47 (1999) 149-157, “Software metrics: successes, failures and new directions” [online]
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.63.2683&rep=rep1&type=pdf
[Accessed 27/10/2019]

GitPrime.”Understanding Code Churn” [online]
https://www.gitprime.com/guides/code-churn/
[Accessed 31/10/2019]

GitPrime. “What is GitPrime Exactly?” [online]
https://help.gitprime.com/general/what-is-gitprime-exactly
[Accessed 2/11/2019]

GitPrime. GitPrime for Executives [online]
https://www.gitprime.com/executives/
[Accessed 2/11/2019]

Hawkey. Dr Kirstie Hawkey, “Software Engineering Processes” [online]
https://web.cs.dal.ca/~hawkey/3130/SEBackground4.pdf
[Accessed 25/10/2019]

Humphrey. Humphrey, W. (2000) “The Personal Software Process” [online]
https://resources.sei.cmu.edu/asset_files/TechnicalReport/2000_005_001_13751.pdf
[Accessed 1/11/2019]

Jira. Jira Software [online]
https://www.atlassian.com/software/jira
[Accessed 2/11/2019]

Johnson, Kou et al. Johnson, P., Kou, H., Agustin, P., Chan, C., Moore, C., Niglani, J., Zhen, S., Doane, W. (2003) “Beyond the Personal Software Process: Metrics collection and analysis for the differently disciplined”
https://www.researchgate.net/publication/4016775_Beyond_the_Personal_Software_Process_Metrics_collection_and_analysis_for_the_differently_disciplined
[Accessed 27/10/2019]

Rosenbaum. Rosenbaum, E. (2019) “IBM artificial intelligence can predict with 95% accuracy which workers are about to quit their jobs” for CNBC [online]
https://www.cnbc.com/2019/04/03/ibm-ai-can-predict-with-95-percent-accuracy-which-employees-will-quit.html
[Accessed 2/11/2019]

SAS. Machine Learning [online]
https://www.sas.com/en_ie/insights/analytics/machine-learning.html
[Accessed 2/11/2019]

Stripe. Stripe.com (2018), “The Developer Coefficient: Software engineering efficiency and its $3 trillion impact on global GDP”, survey [online]
https://stripe.com/files/reports/the-developer-coefficient.pdf
[Accessed 27/10/2019]

Sussman. Sussman, R., Sussman, D. “The future of the professions: how technology will transform the work of human experts” - a discussion on their book ‘The Future of the Professions’ at Oxford Martin School [video]
https://www.youtube.com/watch?v=Dp5_1QPLps0
[Accessed 1/11/2019]

Tangermann. Tangermann, V. (2019) “Amazon Used An AI to Automatically Fire Low-Productivity Workers” for futurism.com [online]
https://futurism.com/amazon-ai-fire-workers
[Accessed 2/11/2019]

Thompson. Thompson, B (2017), GitPrime Blog, “2017 Software Developer Productivity Survey” [online]
https://blog.gitprime.com/2017-software-developer-productivity-survey/
[Accessed 30/10/2019]

Yano, Akitomi et al. Yano, K., Akitomi, T., Ara, K., Watanabe, J., Tsuji, S., Sato, N., Hayakawa, M., Moriwaki, N. (2015) “Measuring Happiness Using Wearable Technology” for ‘Hitachi Review Vol. 64’
http://www.hitachi.com/rev/pdf/2015/r2015_08_116.pdf
[Accessed 25/10/2019]