Ciarán Ingle
15319981

Measuring Software Engineering

Software engineering seems like it would be a trivial subject to measure in comparison to other fields. Software engineers create information, and leave behind them a fully documented virtual trail of all the work they've done in order to achieve their goals. In no other domain is a professional's effect shown through data as much as a software developer.

This visualisation on YouTube shows the development of the Linux kernel from its inception in 1991 until 2015. The clip shows each branch in the project as a node, each being worked on by various developers. As the development cycle ensues, it is clear which developers have the largest impact. Linus Torvalds, the developer primarily credited with creating the Linux operating system can be seen orbiting the project, working on multiple features at once and clearly having the largest impact out of all the other developers. He outshines all other developers in the early stages of the project.

This visualisation would seem like an obvious useful and valid example of measuring impact and analysing software engineering. The truth is; it's **not**.

A huge problem with the video is that all commits made before the adoption of BitKeeper were attributed to Linus Torvalds, regardless of whether or not he was in fact the author. While Linus Torvalds may indeed have had a monumental contribution to the development of Linux, not all of his work was accounted for in this visualisation, and not all of the work he was credited for was in fact done by him.

This illustrates two issues with this model:

1. Software engineers not being credited for work they have done.
2. Software engineers being credited for work they have not done, or had as large an effect as others.

With almost all software development being done in teams nowadays, it is clear that context is important in all forms of analysis and collection of data, and all data needs to be taken into account in order to have an informed opinion on a developer's or team's effectiveness.

The purpose of this report is to discuss some of the current methods of analysing software engineering, and to hopefully determine whether any of these can be applied effectively to current workflows, and at what cost.

A Naïve Approach

There are many basic ways to measure a software engineers effectiveness using non-intrusive methods of public data collection. Let's examine a few of them:

Number of Commits

The most obvious way to determine the number of work a developer does is to monitor the number of commits an author makes to a project. Better developers write good code faster than worse developers, hence they would commit more often.

It is obvious that this assumption has an enormous flaw: *what constitutes a commit?*

Commits can range from inserting a few lines of code to several hundred, but can also include code deletion, merging branches, commenting pre-existing code, and many other options. What is to stop a bad developer from committing broken code in order to receive a higher commit rate and higher pay?

One solution is to impose a build system which prevents commits from being implemented until they pass all of the test. But what if a developer commits working code in small chunks of code at a time? The solution is obvious:

Number of Lines

The number of lines committed could be a valid way of assessing an engineer's work. Better developers write more lines of good code than bad developers. But yet again, this solution is not so effective.

Consider this example: Developer one has written a for loop, but developer two writes the same function using a while loop. Obviously a while loop occupies more lines of code than a for loop even though they may have the same functionality.

Clarity and brevity are not rewarded in this system of analysis. Instead of encouraging efficient code, developers would be incentivised to write longer functions in order to achieve the same result. This would in fact hinder a project's development.

Would a developer removing redundant code from a project be penalised? We need to find a different solution entirely.

Advanced Data Collection

It is clear from the previous naïve solutions that analysing easily accessible data does not yield an effective insight into the software development process. For advanced analysis we need more expansive and more detailed data sets.

According to Philip M. Johnson's paper 'Searching under the Streetlight for Useful Software Analytics', "Developers and researchers must weigh the trade-off between easily obtained analytics and richer analytics with privacy and overhead concerns."

He argues that the easier an analytic is to collect and the less controversial it is to use, the more limited its usefulness and generality. Therefore, in order to collect more insightful data, we need to use more advanced tools and processes to give us access to rich, high-impact analytics.

The Personal Software Process (PSP)

"The PSP is a structured software development process that is intended to help software engineers better understand their performance by tracking their predicted and actual development of their code." -*Wikipedia*

The aims of the Personal Software Process are to help engineers:

1. Improve their planning skills.
2. Make commitments they are capable of fulfilling.
3. Manage the quality of their work.
4. Reduce the amount of defects in their projects.

The original version of PSP used simple spreadsheets and manual data collection for manual analysis. Developers had to fill out 12 forms containing 500 distinct values, which required substantial effort. Some of these forms included a project plan summary, a time recording log, and a design checklist.

Watts Humphrey, the creator of the Personal Software Process, approved of the manual nature of the process as he claimed that it required good judgement in the collection of data. He encouraged developers to modify and expand on the process as they saw fit.

Obviously, the manual nature of this process makes the PSP a redundant method of data collection in present day. This is why tools like the Leap Toolkit were introduced.

The Leap Toolkit

This tool attempted to rectify the problems with the quality of data being collected by its successor, through automating and normalising the data analysis stage. This hybrid process still requires significant manual data input however. The Leap data is portable and maintains data about a single developer at all times as they move between projects.

Removing the manual data analysis makes the Leap toolkit much less of a burden than PSP, but is still not ideal. A fully automated approach would be better.

Hackystat

The Hackystat framework was created in order to fully automate the data collection process for developers. It is used for collection, analysis, visualisation, interpretation, annotation and dissemination for the software engineering process. The framework has little to no overhead for developers, as sensors have to be implemented into development tools in order to gain insight.

Integrating this project into developer's work days included creating editors, build tools and testing tools for client side data collection. Events such as switching files, editing methods, constructing test cases, and invoking tests were all monitored by this software.

Hackystat was invented with four design features in mind:

1. Client & server side data collection.
2. Unobtrusive and non-disruptive collection of data.
3. Minute-by-minute collection of data.
4. Personal and group-based development.

The software ICU gives indications to a developers statistics, such as:

- **Dev Time** - How much time was spent in the developer's IDE working on each file associated with the project.
- **Commit** - How often the developer committed to the repository along with the number of lines of code.
- **Build** - How many times the developer built the system and whether that build was successful.
- **Test** - How often each developer invoked the test suite and the success rate of those tests.

Ciarán Ingle
15319981

Big Data Analysis

More thorough and intensive data collection methods create much larger datasets than basic analytics. Building the infrastructure needed for big data analysis can be quite a large expense for companies, especially if a company does not have the same resources available to them as larger companies. It is much cheaper and more insightful to outsource data analysis to dedicated services.

For this purpose, many companies have devoted their existence and speciality to analysing and providing visualisations of their clients' big data.

Google Cloud Platform

One example of a company that offers analytics services is Google. Google's Cloud Platform offers a plethora of services to companies, one aspect of these being data storage and data analysis.

Google Cloud Datalab is an interactive tool that can be used to analyse and visualise data. It builds machine learning models from the input data set that run on the Google Compute Engine. It queries terabytes of data in BigQuery, and runs local analysis on sampled data in the Cloud Machine Learning Engine. The Machine Learning Engine allows businesses to build machine learning models on any type of data, and utilises deep learning.

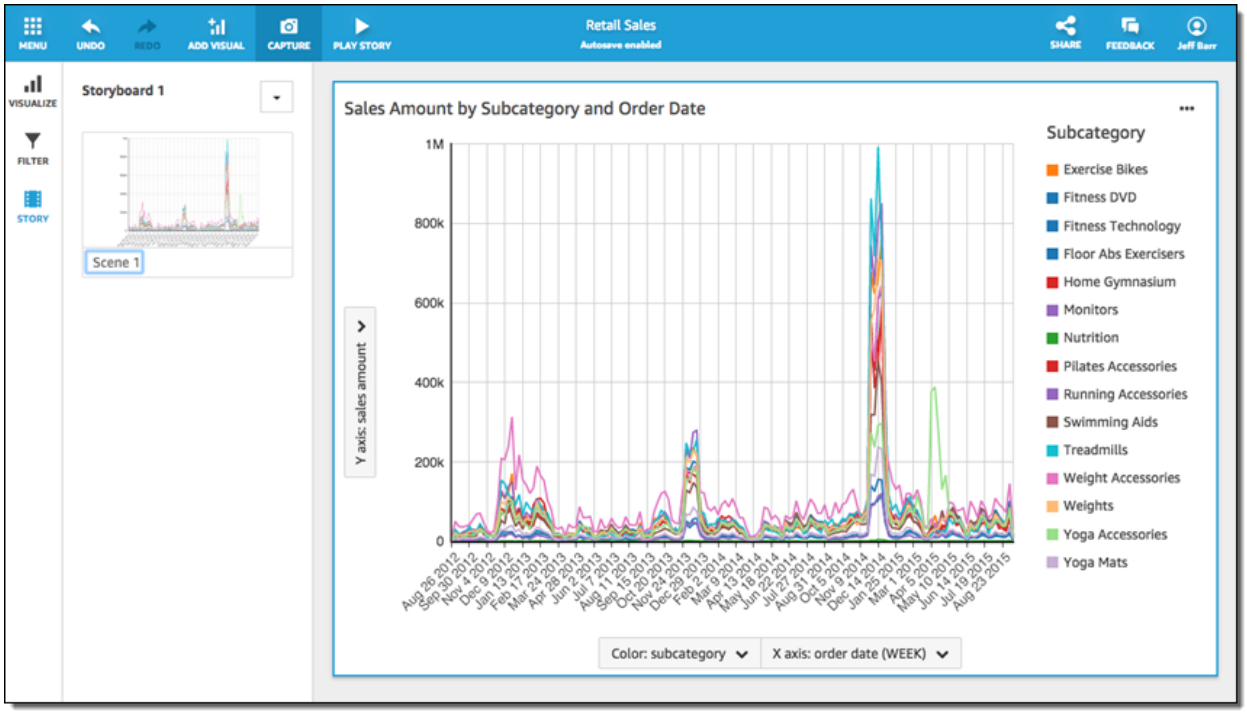
Using a trainer package and data, a business can begin training the Cloud ML Engine model. The service allocates resources in the cloud according to the job request's specifications. During training, a trainer writes regular checkpoints and sends regular logging information before exporting the model at the end of the job. Cloud ML Engine supports both online and batch learning.

Ciarán Ingle
15319981

Amazon Web Services

Amazon also has also developed technologies for AWS that can work with massive data efficiently, scalably and cost effectively. They offer services like data warehousing, batch processing and machine learning.

One such product they offer is Amazon Quicksight which is used for business analytics, and claims it costs 1/10th the price of traditional BI solution. The cloud-powered service builds visualisations and performs ad-hoc analysis from the client's data from any browser or mobile device.



The client points the service to a data source in a variety of locations, such as a CSV file, a SQL database or a number of other applications, and can immediately begin creating visualisations of this data. The SPICE engine uses columnar storage, machine code generation and data compression to allow for interactive queries and rapid response times.

Insights can be easily implemented into business dashboards to be shared with other entities in the organisation.

Ciarán Ingle
15319981

Ethics of Data Collection & Analysis

When collecting and operating on data that is not publicly available, some discomfort must be introduced to the subjects of the analysis. More than often with these advanced forms of data collection, developers must put up with intrusive or distracting methods of data collection in order for companies to gain insight into their workflow.

It's easy to see why companies want to gain such detailed insight into their employees and customers. According to the McKinsey Global Survey Results, 'Evolution of the Networked Enterprise', the benefits are more than worth the risks. Respondents to the survey claimed that these social tools contribute to 20 percent of revenue increases and 18 percent of cost improvements. "A majority say their companies' leaders see leaks of confidential information as a significant risk related to social tools, yet 60 percent agree that the benefits far outweigh the risks."

Privacy

Again, to reference a key quote from Philip M. Johnson's paper, "Developers and researchers must weigh the trade-off between easily obtained analytics and richer analytics with privacy and overhead concerns."

This tradeoff was apparent with initial feedback to the introduction of the HackyStat toolkit described earlier. Developers complained about the automated nature of the data collection, and were unhappy with the invasion of their privacy necessary to gain insight into their workflow. Software engineers didn't want to install software that would collect data about their activities and without their knowledge of what was being submitted.

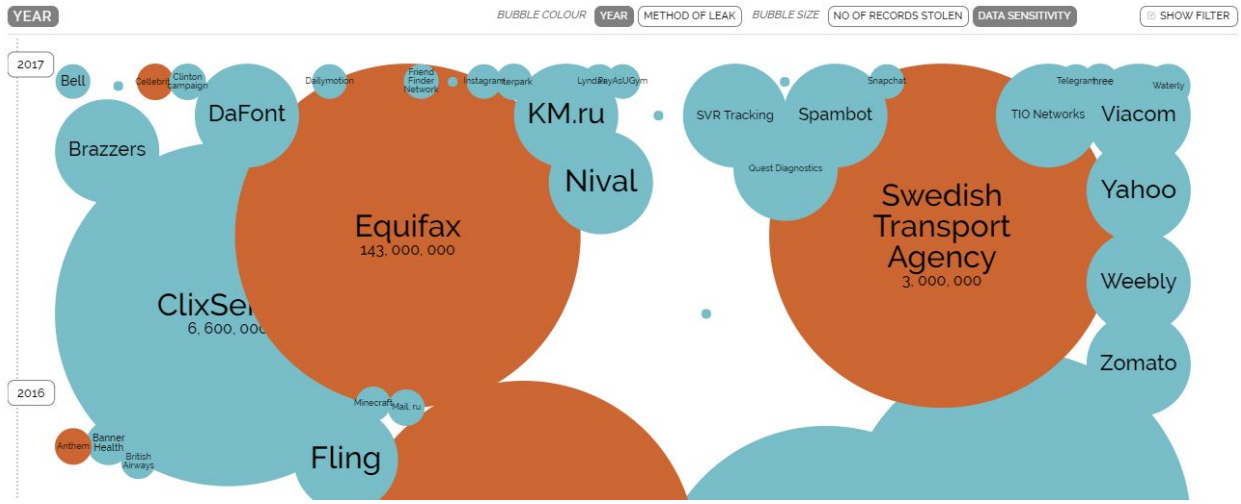
The term "Hacky-stalk" was used by one developer to describe the software, due to its transparency it provided for development groups, along with each group member's work style. Developers repeatedly informed the Hackystat creators that they were uncomfortable with the amount of data management were being given access to, despite promises of appropriate usage and restraint by superiors.

It is understandable that developers would be worried that certain information could give rise to being let go or otherwise being punished by their company.

Security

Another liability of more advanced forms of data collection is guaranteeing the security of private information. The more data being collected about employees, the more data there is to be leaked through a potential security hole.

According to Information Is Beautiful's 'World's Biggest Data Breaches', large data breaches are being committed at a worrying in rate. In 2017 alone, companies such as Equifax, ClixSense and the Swedish Transport Agency have had enormous data leaks involving millions of customers and employees each.



In the case of the Boeing, 36,000 employees' details were leaked after data left control of the company due to negligence on the part of one employee. Even if data is not being explicitly stolen, it is still important that the data being collected is secure from even those in the company.

Such measures as encryption, firewalls, authentication and proper training become more important as the size and detail of data collection increases.

Conclusion

“Developers and researchers must weigh the trade-off between easily obtained analytics and richer analytics with privacy and overhead concerns.”

This is the key point in any argument to be had over the ethics of data collection and analysis. As more companies see the revenue increases and productivity benefits due to more invasive data collection methods, tools such as “Hacky-stalk” are going to become more and more prevalent in the average modern workplace.

With young aspiring developers in such large and frequent supply, it will be no surprise when companies start to give resistant employees ultimatums between job security and personal privacy. Younger developers who are already more comfortable with the level of data collection present in their daily lives with their smartphones, Fitbits, social media and online shopping will no doubt be less vocal about privacy invasion at work.

The future of data collection and analysis will surely come in the form of unnoticeable products such as software extensions, or indeed hardware wearables attached to every employee such as badges or identification cards. With many modern offices adopting Google’s ‘Work as a playground’ approach to office design, employee freedom and privacy is already on its way out the door.

According to Humanyze, the company designing employee-monitoring badges, "It's powerful to see how people want to display better behaviours or the behaviours that you're moving them towards." Their viewpoint is that employee productivity is driven by feedback, so in essence the monitoring in fact improves the company’s workflow.

Ultimately, employee feedback will be the only thing capable of stopping the completely Orwellian office culture, but in the current climate, such a resistance seems unlikely from the younger upcoming generation of graduate software engineers.

Sources

The Personal Software Process:
https://en.wikipedia.org/wiki/Personal_software_process

Searching under the Streetlight for Useful Software Analytics:
http://www.citeulike.org/group/3370/article/12458067

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

The Hackystat Framework:
http://csdl.ics.hawaii.edu/research/hackystat/

Google Cloud Platform:
https://cloud.google.com/products/big-data/

Google Machine Learning Engine:
https://cloud.google.com/ml-engine/

AWS Data Analytics:
https://aws.amazon.com/products/analytics/

Largest Data Breaches:
http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/

How Data Collection Technology might change Office Culture:
http://www.cbc.ca/news/technology/how-new-data-collection-technology-might-change-office-culture-1.3196065