Software Engineering
Prof. STEPHEN BARRETT

REPORT

Measuring Software Engineering

—

By Satwik Chandra (19321427)

Index

OBJECTIVE

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 ethical concerns surrounding this kind of analytics.

Software measurement

Software measurement is a vital asset in the software engineering discipline. Professionals involved in software engineering are often expected to keep up with new technologies and also with the highly competitive market. On top of that, they also have to focus on the reliability of the product, the stability of the product and also the testing of the product. Hence to discover new development techniques and optimise the production time and quality, there is a strong need for a way to measure engineering.

Software measurement can help a software engineer, increase return on investment (ROI), identify areas of improvement, manage workloads, reduce overtime, and reduce costs. [1]

There are many different types of metrics available to measure software engineering.

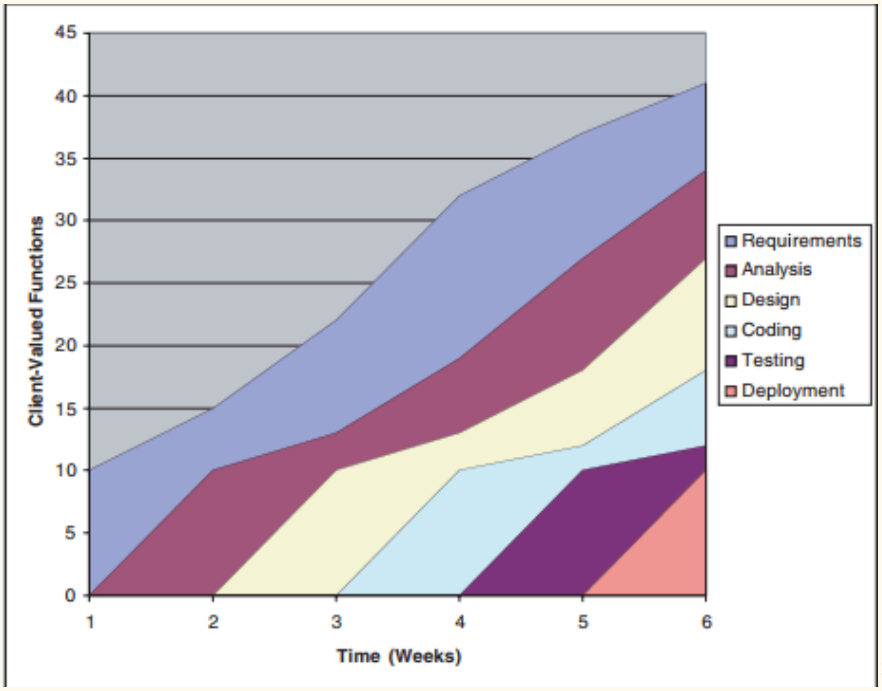
Some of them are as follows:

Lines of Code(LOC)

Also known as **source lines of code(SLOC)**, LOC is a software metric that is used to measure the size of a computer program by counting the number of text lines in the program's source code. LOC is mostly used as an indicator for programmers to see how much work is left to be done for the program to be complete. This metric is also used to estimate the productivity and maintainability of the program once the software has been produced[2]

Lead Time

Lead time quantifies how long it takes for ideas to be developed and delivered as software. Lowering lead time is a way to improve how responsive software developers are to customers.



[3]

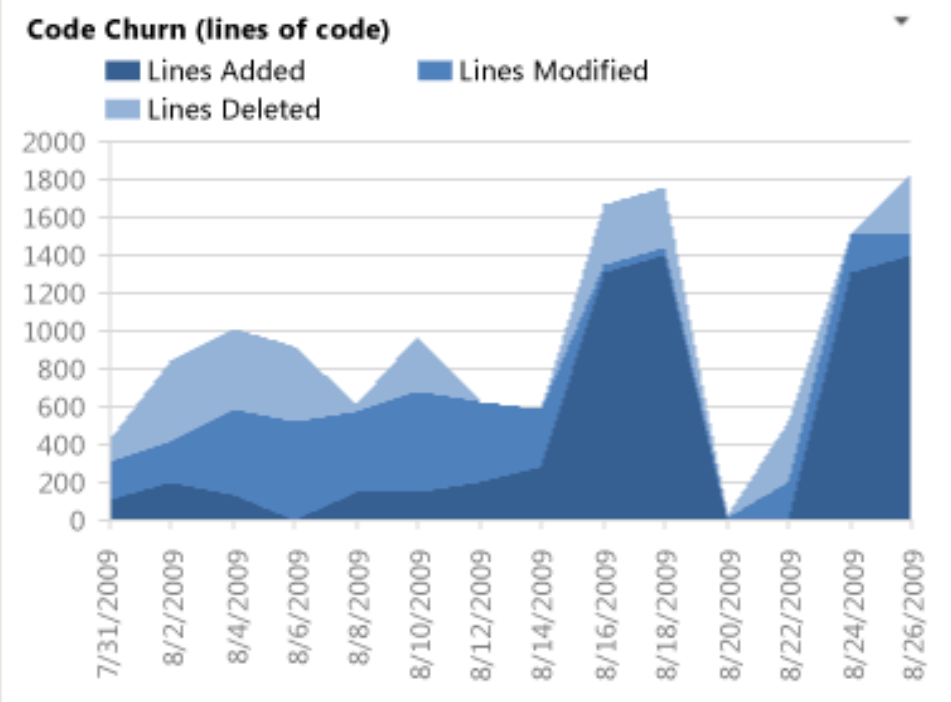
[1] https://stackify.com/track-software-metrics/
[2] Source lines of code - Wikipedia
[3] http://catalogue.pearsoned.co.uk/samplechapter/0131424602.pdf

Cycle Time

Cycle time describes how long it takes to change the software system and implement that change in production.[4]

Code churn

Code churn represents the number of lines of code that were modified, added or deleted in a specified period of time. If code churn increases, then it could be a sign that the software development project needs attention.[5]

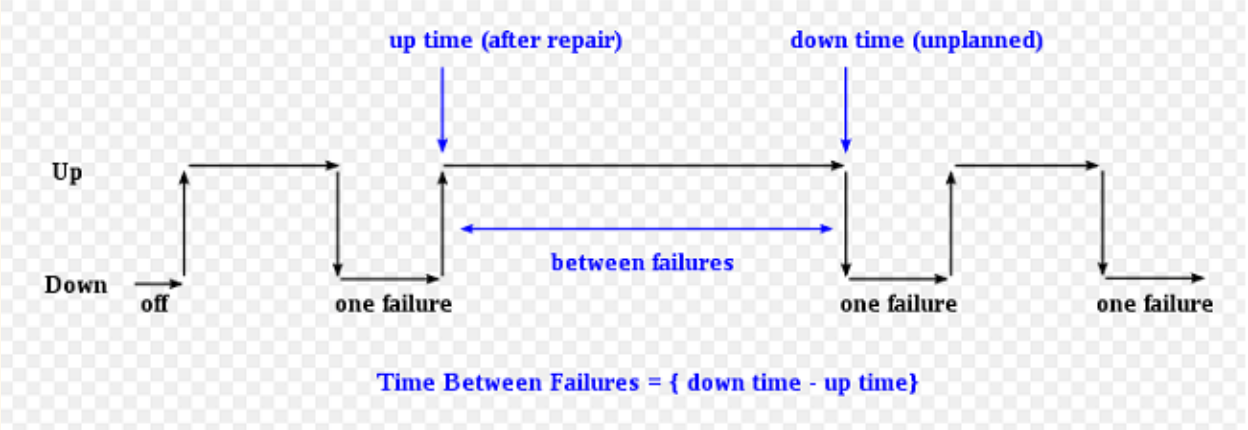


[6]

[4] http://screenful.com/blog/software-development-metrics-cycle-time
[5] https://blogs.msdn.microsoft.com/askburton/2004/09/09/q-what-is-code-churn/
[6]https://www.visualstudio.com/en-us/docs/report/sql-reports/perspective-code-analyze-report-code-churn-coverage

Mean time between failures(MTBF) and mean time to recover/repair(MTTR)

Both metrics measure how the software performs in the production environment. Since software failures are almost unavoidable, these software metrics attempt to quantify how well the software recovers and preserves data.



[7]

Application crash rate(ACR)

Application crash rate[8] is calculated by dividing how many times an application fails(F) by how many times it is used(U).

ACR = F/U

Computational programs

Leap

Leap toolkit is created to provide Lightweight, Empirical, Anti-measurement dysfunction, and Portable approaches to software developer improvements. If software engineers use Leap they can gather and analyze personal data regarding time, size, defects, patterns, and checklists.

In the platform Leap, there are two main activities. These are gathering primary data and performing Leap analyses. These can be increased by secondary activities of refining

[7]https://commons.wikimedia.org/wiki/File:Time_between_failures.svg
[8]https://www.apteligent.com/technical-resource/best-practice-2-maintain-a-crash-rate-of-less-than-0-25-in-your-apps-three-most-critical-userflows/

definitions, checklists, and patterns. Finally, these central and/or secondary activities can be directed toward individual skill acquisition and improvement or group review of work products.[9]

Hackystat

Hackystat is an open source framework that automatically collects data and analyses the data from the product and software engineers. The aim is to provide a mechanism that is extendable and has the ability to reduce the overhead associated with collection of a wide variety of software engineering data. The platform also provides a toolkit of analysis techniques which can be used to create a useful report.

Use of Machine learning on Software Engineering metrics

We discussed before how having metrics in software engineering can help us optimize production time and quality.

There are various computational tools and techniques available to use on the data collected from the metrics discussed above to optimize the production quality. In software development, the quality of the development process is important for the quality of software products. A high quality development process leads to high- quality products. Hence one of the computational techniques we can use to achieve that is using machine learning[10].

In machine learning technique, software metric data is used as input for learning algorithms. The first aspect of this technique is the learning and verification of thresholds of software metrics and the analysis of metric sets. This can be used to efficiently locate problematic sections of source code. The second is to analyse software development processes. For this, metric data measured at different points of time during the execution of software projects is used.

Software metrics are often used in combination with thresholds. If the value of a metric violates a threshold, some kind of problem with the measured entity is indicated. With this information it is possible to differentiate between good and bad entities. The metric values measured over an entity can be interpreted as a vector in the n-dimensional real-space. The combination of the classification in good and bad entities with their corresponding metric values results in a **supervised** learning sample.

---

[9] Leap: a "personal information environment" for software engineers | Semantic Scholar
[10] https://www.swe.informatik.uni-goettingen.de/sites/default/files/publications/research_summary.pdf

An algorithm to learn n-dimensional rectangles can be applied to such a learning sample. The algorithm determines a rectangle, such that the good entities are inside the rectangle and the bad ones outside. The borders of the rectangle are interpreted as the thresholds. This way the original thresholds that were used to classify the entities can be verified.

There are many possible software metrics that can be utilized for such an algorithm, depending on the exact application. Some examples are the size of specification documents, the lines of code, the lines of comments, test case coverage, reported number of bugs, or the passed and failed test cases. All of these metrics represent certain features of a project, for example the size of a product. How the values of these develop, should depend on the current phase of a project. For example, the size of the specification documents should only change considerably during the design phase.

A general problem is the quality of the data: Since projects are usually imperfect, the data will be imperfect as well. By using learning algorithms that can cope with noise, we implicitly solve this problem. The imperfectness of the data results in noise.

Ethical concerns regarding collection and use of software engineering metrics.

In retrospect, the same ethical concerns apply to use of machine learning on software engineering metrics as it would with using machine learning on any other industry area. The ethics regarding collection of data to be used as input for the learning algorithms are similar as well. On top of that, algorithmic bias also creeps into the analysis of the data collected.

Some concerns can be that data might be collected without the consent of the developer,data collected and analysis of that data could result in an unbalanced power relationship between the project manager and the developer team.

Getting software development process data from every developer’s computer can be unethical if there is no formal consent given before.

Furthermore this method of data collection could add to the stress of developers to satisfy a certain metric and this stress could further negatively affect production time and quality but most importantly a safe work environment.

Protection of collected data could also be an added concern, since the data collected is private data and could also contain confidential information about the developer team. The data collection methods can also be biased because of the environment, the personal opinions of the project manager to favour one metric to another.

Although all the statements above are just concerns which can be dismissed by applying necessary safeguards to the data collection process and following the ethical code of conduct for data collection[11].

If the necessary safeguards are met and the data collection protocols are followed appropriately, collection of software engineering metrics could in fact help with the development process, resulting in increase of quality of both the development phase and also the product. Metrics if used correctly could also help optimize the development process while also allowing to reduce overtime and "crunching[12]" in the development phase.

Conclusion

The report aims to find the different metrics using which we can measure software engineering process, from calculating the lines of code, code churn, Application crash rate etc. These metrics help a software manager to better manage his or her team resulting in optimising the production time and also reducing overtime for the development team. Some computational programs are indeed available in the industry, like Leap, to analyse these metrics but although analysis and collection of this data does introduce a cause for some ethical concerns; if there are necessary safeguards, protecting the data and the provider of the data, these metrics can in fact help optimise the development process and reduce overtime and crunching which can help reduce the stress and workload of the development team but also help produce a well produced deliverable.

[11] https://www.njhealthmatters.org/content/sites/njhc/Ethical_Considerations_in_Data_Collection.pdf
[12]https://cs.stanford.edu/people/eroberts/cs181/projects/crunchmode/index.html#:~:text=%22Crunch%20mode%22%2C%20also%20referred,project%20or%20meet%20a%20deadline.

Bibliography

https://stackify.com/track-software-metrics/

https://blogs.msdn.microsoft.com/askburton/2004/09/09/q-what-is-code-churn/

http://screenful.com/blog/software-development-metrics-cycle-time

Source lines of code - Wikipedia

http://catalogue.pearsoned.co.uk/samplechapter/0131424602.pdf

https://www.visualstudio.com/en-us/docs/report/sql-reports/perspective-code-analyze-report-code-churn-coverage

https://commons.wikimedia.org/wiki/File:Time_between_failures.svg

https://www.apteligent.com/technical-resource/best-practice-2-maintain-a-crash-rate-of-less-than-0-25-in-your-apps-three-most-critical-userflows/

Leap: a "personal information environment" for software engineers | Semantic Scholar

https://www.swe.informatik.uni-goettingen.de/sites/default/files/publications/research_summary.pdf

https://www.njhealthmatters.org/content/sites/njhc/Ethical_Considerations_in_Data_Collection.pdf

https://cs.stanford.edu/people/eroberts/cs181/projects/crunchmode/index.html#:~:text=%22Crunch%20mode%22%2C%20also%20referred,project%20or%20meet%20a%20deadline.