Measuring Software Engineering

Caolan Wall 17329660

Introduction

Efficient software engineering is one of the best ways for a company to become one of the best performers in their industry. Therefore, engineering efficiency has become one of the most important topics in IT departments across the globe. Companies invest huge capital into their software engineering process and so it is important for them to have metric that they can use to gauge how their process is coming along.

The best software engineering teams will put a huge emphasis on improving their process and their output. They will keep track of a certain set of indicators known as metrics or Key Performance Indicators (KPIs).

Measuring and Assessment of Software Engineering

Software Measurement uses quantifiable attributes to determine quality of the processes of software engineering. Software engineering metrics should be simple to compute, consistent and unambiguous, independent of programming languages, easy to calibrate and adapt.

Keeping track of these metrics paints an overall picture of how the software development process is going, showing if an area needs to be improved or if the process is on track. Metrics are an important component of quality assurance, debugging, performance, and estimating costs. Here are some common metrics used in the industry:

Source Lines of Code

- Source Lines of Code or SLOC is a measurement used in Software Engineering to determine the productivity of programmer. It can also be used to predict the effort required to develop a program.
- This has been used since the 1960's however, this can encourage copy and pasting code and discourage refactoring of code after it is written.

Code Churn

- This is a more sophisticated analysis of the source lines committed.
- This measure the lines of code that are discarded or replaced with each commit during the development process.
- A high level of code churn can show signs of a team or individual who are/is uncertain of the requirements of their project or how they will go about developing their solution.
- Code churn is to be expected in software development however if the rate is too high it can be a sign of a weakness in the development team.

Mean Time Between Failures

- This refers to the average time that the program will run for before failing.
- This is used to predict how likely the program is to fail in a certain period of time.

Mean Time to Repair

- This is the time taken to restore a system to its full functionality.
- The clock starts when the repair starts and includes the testing period until the system is restored.

Mean Time to Recover

- This is similar to mean time to recovery, but it includes the failure notification time as well as diagnosis.

McCabe's Cyclomatic Number

- This quantifies this complexity of a program by measuring the number of linearly independent paths through the program, - the higher the number the more complex it is.
- It uses a flow graph to determine the complexity.
- The formula is CC = E – N + 2P, where E is the number of edges (transfers of control), N is the number of nodes (sequential group of statements with only one transfer of control), and P is the number of disconnected parts of the flow graph (calling a subroutine).

Velocity

- Velocity is a metric used in Agile development models and is used to determine the rate of progress for a development team.
- It is the measurement of work done over a time period.
- The product is broken down into components called user stories which are how a product will deliver back to the user.
- The unit of time is called a sprint.
- The number of user stories a team can deliver in one sprint is the velocity of development.
- It is not useful to compare team's velocities as the metric is based on non-objective estimates.

Opens/Close rates

- This is how many production issues are reported and closed within a specific time period.
- The average of open and close rates is more important than the actual numbers

Time Active

- This is a very basic metric which is the measure of how much time a team spends coding and does not include planning and other tasks.
- However, this is a vague metric when used on its own as it does not account for the complexity of the project at hand.

Leadtime

- This is how long it takes to go from an idea to delivering the software.
- A smaller leadtime mean happier customers and more time for other projects.
- Leadtimes are typically reduced by simplifying decision making and reducing waiting time.

Cycle time

- Cycle time a subset of the development process.
- A subset could be implementing a bug fix or making changes to the functionality.
- It is useful to monitor cycle time as it allows you to establish a baseline for tasks so if a task takes longer you can address it.

Endpoint Incidences

- This is a measure of the security of a project and is an easy and insightful analysis.
- This is the number of times a device running the program has been attacked by a virus or other malicious software.

Computational Platforms to Measure the Software Engineering Process

GitHub

GitHub is a Git repository hosting service, but it adds many of its own features. While Git is a command line tool, GitHub provides a Web-based graphical interface. GitHub is the most widely used version control system. It has over 37 million developers and over 100 million repositories, 28 million of which are public.

GitHub allows you to keep track of a codebase but also provides tools to analyse the metrics of the code base. GitHub has a version control system, allowing developers to manage and store revisions of their projects. It also allows teams to collaborate and work off the same codebase.

A board member of GitHub, Peter Levine, once wrote in a blog post "*Source Code Management (SCM) is the second most fundamental tool for a programmer after compiler and development tools. It stores, versions and branches source code being developed by teams of programmers. At scale, these systems become highly complex and often difficult to manage. In addition, historically SCMs have been anti-social. The No. 1 conversation they generate is referred to as: "Who broke the build?" GitHub solves these two problems and dramatically expands the category by changing the old model in two important ways:*

Rather than forcing every development team in the world to deploy their own SCM, GitHub runs one big SCM in the cloud and the management issues vanish.

GitHub organizes projects around people rather than code."

This is an insight into the problem's developers had run into in the past using basic source code management.

Forking is one of GitHub's main selling points. It allows users to copy a repository from one user to another, make changes and then "push" those changes so other users can "pull" this code and edit it.

Each time a developer commits code to the repository, its time is logged along with a comment from the developer. It also automatically shows the Source Lines of Code and the Code Churn, mentioned above.

Github insights is a useful way for a project manager to gauge the effort each member of his team is putting into their project.

Codacy

Codacy is perhaps the most widely used computational platform in software engineering analysis and is incorporated into popular software version control platforms such as Github.

It provides automated analysis of uploaded code including useful metrics such as cyclomatic complexity, code coverage and duplication. These can in turn be used to make important judgements on the performance of a team and the quality of the code they produce.

Algorithmic Approach

The metrics we collect can be more insightful when we put them into certain algorithms. These algorithms calculate specific attributes of the software engineering process and give a greater understanding of the process's downfalls.

COCOMO Model

The Constructive Cost Model is a procedural cost estimate model for software projects and often used as a process of predicting the various parameters associated with making a project such as size, effort, cost, time. It was proposed by Barry Boehm in 1970 and is based on the study of 63 projects, which make it one of the best-documented models. It is based on the aforementioned SLOC.

Putnam Model

The Putnam model describes the time and effort required to finish a software engineering project of a certain size.

$$\mathbf{L = C_k K^{1/3} t_d^{4/3}}$$

This is the equation used for the Putnam model where K is the effort in development, L is the product size in LOC, t_d is the time, and C_k is the state of technology constant.

As you can see, when the time schedule of a project Is reduced this leads to a big increase in the required effort and cost.

Agile

Agile software development refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between teams. Agile methods generally promote a disciplined project management process which encourages good leadership, good teamwork, self-organisation and accountability, good practices and combining customer needs with company goals.

Scrum and Kanban are two of the most widely used Agile methodologies.

Scrum

Scrum is a framework that helps teams work together. Scrum encourages teams to learn through experiences, self-organize while working on a problem, and reflect on their wins and losses to continuously improve. Scrum describes a set of meetings, tools, and roles that work together to help teams' structure and manage their work.

There are three main artefacts in scrum, product backlog, sprint backlog and an increment.

Product Backlog is the list of work that must be completed maintained by the product owner or product manager. This is a dynamic list of features, requirements, enhancements, and fixes that acts as the input for the sprint backlog.

Sprint backlog is the list of items, user stories, bug fixes that are selected by the development team for the current sprint cycle. A team must choose which items it will work on during the sprint before they begin.

Increment is the usable end product from a sprint. It is when the goal of a sprint is completed or "done". The definition of done varies from team to team, it could mean the product is released to the customer or finishing a part of a system that is not yet ready to be shipped.

A sprint is the actual time period when the scrum team works together to finish an increment. Two weeks is a pretty typical length for a sprint, although some teams find a week to be easier to scope or a month to be easier to deliver a valuable increment.

At the end of a sprint, the team will get together for sprint review. The team will showcase the product backlog items that are now "done" to teammates and stakeholders.

Waterfall Model

This was first introduced by Dr. Winston W. Royce in 1970 and was the first process model to be introduced. Each phase must be completed before the next phase can begin and there is no overlapping in the phases.

1. Requirements: The first phase involves the requirements of the project. Here, the specifications of the input and output or the final product are studied and written down into a requirements document.
2. System Design: The requirement specifications from the first phase are studied in and system design is prepared. System Design helps in specifying system requirements and also helps in defining overall system architecture.
3. Implementation: With the system design in mind, the system is first developed in small programs called units, which are integrated into the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
4. Integration and Testing: All the units developed in the implementation phase are integrated into a system after testing of each unit. Constant software testing is required to find out if there are any flaw or errors. Testing is done so that the client does not face any problem during the installation of the software.
5. Deployment of System: Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.
6. Maintenance: This step occurs after installation and involves making modifications to the system or an individual component to alter attributes or improve performance. These modifications arise either due to change requests initiated by the customer, or defects or bugs uncovered during live use of the system. The client is provided with regular maintenance and support for the developed software.

The waterfall model is easy to use as it is simple to follow the progress as there are no overlapping stages. The Waterfall model works well for smaller projects where requirements are very well understood.

Ethical Concerns

When it comes to software engineering the specific ethical concerns involved are dependent on the project however there are general outlines every software developer should use.

According to IEEE-CS/ACM Joint Task Force on Software Engineering Ethics and Professional Practices, software engineers should adhere to the following principles.

1. Public – Software engineers shall act consistently with the public interest.

2. Client and employer – Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.

3. Product – Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.

4. Judgment – Software engineers shall maintain integrity and independence in their professional judgment.

5. Management – Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.

6. Profession – Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

7. Colleagues – Software engineers shall be fair to and supportive of their colleagues.

8. Self – Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Data Protection

A major and very topical concern now is data protection. Personal data security is one of the biggest concerns in the digital world because of the sensitive information that clients trust software engineers with. Personal information is a point of interest for many organizations, from national security to cybercriminals.

The EU has made huge strides in protecting users with their General Data Protection Regulation (GDPR). GDPR was approved by the EU Parliament on 14 April 2016. It was enforced on 25 May 2018 and organisations that are not compliant could now face heavy fines. It is designed to harmonize data privacy laws across Europe, protect and

empower all EU citizens data privacy, and reshape the way organizations across the region approach data privacy.

Loss of Life

The safety of software systems when a potential of loss of life is involved is paramount. We recently saw the crash of two of Boeing's brand-new Boeing 737 Max 8 leading to the deaths of 346 passengers and crew.

The cause of the crash of Lion Air Flight 610 and Ethiopian Airlines Flight 302 was caused by the Manoeuvring Characteristics Augmentation System (MCAS). MCAS is a system which was designed to mitigate the aircraft's tendency to pitch up compared to the other models of Boeing 737 aircraft due to the new engines the Max 8 comes equipped with. The system was implemented to keep the Max 8 in the same class of plane as existing Boeing 737's, allowing pilots to jump between aircraft with no additional training required, a huge selling point for airlines.

The MCAS on these two planes received false warnings of a high angle of attack, forcing the plane's nose down and ultimately into the ground. Pilots were not trained on how to disable this system although it is possible to do so.

The fleet of 387 Max 8s across 59 airlines has now been grounded since the 13th of March this year.

From the reports available, it is clear Boeings corporate greed was ultimately the cause of these accidents. Additional safety systems which could have fed other information to the MCAS and accounted for the misreadings of the angle of attack were scrapped in favour of profit.

Conclusion

Software engineering is big business and is now subject to the production disciplines that were first implemented in manufacturing plants under a lean process regime. In modern software companies' metrics are applied to the software engineering process in order to measure the efficiency and determine where improvements can be made. To that end computational platforms have evolved and been incorporated into the software engineering tool set. In addition, algorithms have been developed to assist in measuring the efficiencies and provides a constant loop feedback.

Software engineering processes were largely based on the waterfall method which relies on accurate requirements being identified up front. In recent times Agile Methodologies have been introduced which allow for changing requirements through time limited development periods called Sprints. Agile recognises that companies can change direction and the software engineering process must align itself accordingly.

Software Engineering also incorporates ethical responsibilities on the people involved as if the products delivered does not work as designed there can be catastrophic results including loss of life.

Bibliography

http://www.chambers.com.au/glossary/mc_cabe_cyclomatic_complexity.php

https://techbeacon.com/app-dev-testing/9-metrics-can-make-difference-todays-software-development-teams

https://www.scaledagileframework.com/

https://searchsoftwarequality.techtarget.com/definition/user-story

https://techcrunch.com/2012/07/09/github-pours-energies-into-enterprise-raises-100-million-from-power-vc-andreesen-horowitz/

https://www.dummies.com/careers/project-management/agile-project-management-for-dummies-cheat-sheet/

https://www.youtube.com/watch?v=2Vt7Ik8Ublw

https://www.nytimes.com/2019/10/02/business/boeing-737-max-crashes.html