Measuring Software Engineering Report

CS3012

Divine Mbunga

17324651

28/10/2019

Table of Contents

Abstract

This report discusses the ways in which the software engineering process can 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.

History of Software Engineering

Software engineering is the process of analysing user needs and designing, constructing, and testing end user applications that will satisfy these needs using software programming languages. It is application of engineering principles to software development. (1)

In early 1967, there was an awareness of the rapidly increasing importance of software systems in many activities of the society. The main driving factors that lead to the organisation of the first conference on software engineering in 1968 was the result of many problems faced in the software manufacturing industry causing a belief that techniques should become less ad hoc, and instead be based on theoretical foundations and practical disciplines that are established in traditional branches of engineering. The goal of this conference was “the establishment and use of sound engineering principles in order to obtain reliable, efficient and economically viable software”. This was the beginning of the software evolution. Maintenance was considered as a post-production activity, and that inspired Royce to propose the waterfall life cycle process. The maintenance is the final step like bug fixes and minor adjustments, to the development of software. This process is a classical view on software engineering and is still in use today by many companies. It took time for the software engineers to realise the limitations to this software process model, that the separation in phases was too strict and inflexible, and that it was unfeasible to assume that the requirements are known and definite at the beginning of the software design process. In the seventies an attempt towards a more evolutionary process model was proposed by Yau known as the change mini cycle. Since then, more process models have been proposed. All processes involve specification of the system, the design and implementation, validation of the system and evolution by changing the system in response

to needs. It took until the nineties for the term software evolution to have a more widespread acceptance with research on the topic to be popular. (2)

Measurable Data

The workplace of software engineering involves rapid development and delivery making it impossible to produce a set of stable software requirements. In order to keep up with the fast – paced industry, it is important for the engineering team and their managers need to have a clear idea of their performance in terms of the project. Questions concerning completion time, effort needed, quality of code and costs need to be answered. Thus, the management team needs to have access to measurable data that can be used to answer these questions. To evaluate the progress and productivity of the team, there are several methods to be considered.

Agile methods

Agile methods are based on teamwork, customer collaboration, iterative development and people, process and technology adaptable to change. Agile methods are new product development processes that can bring innovative products to market quickly and inexpensively on complex projects with ill-defined requirements. Agile methods are based on four values derived from the Agile Manifesto:

1. Individuals and interactions
2. Customer collaboration
3. Working software
4. Response to change.

Examples of agile methods that include these values are Scrum, Extreme Programming, Feature Driven Development, Dynamic Systems Development, Pair Programming and Crystal Clear. With communicating regularly, involving the customer, building working software and being adaptable to changing needs, products can be developed and delivered inexpensively, quickly and in good quality. (3)

(4)

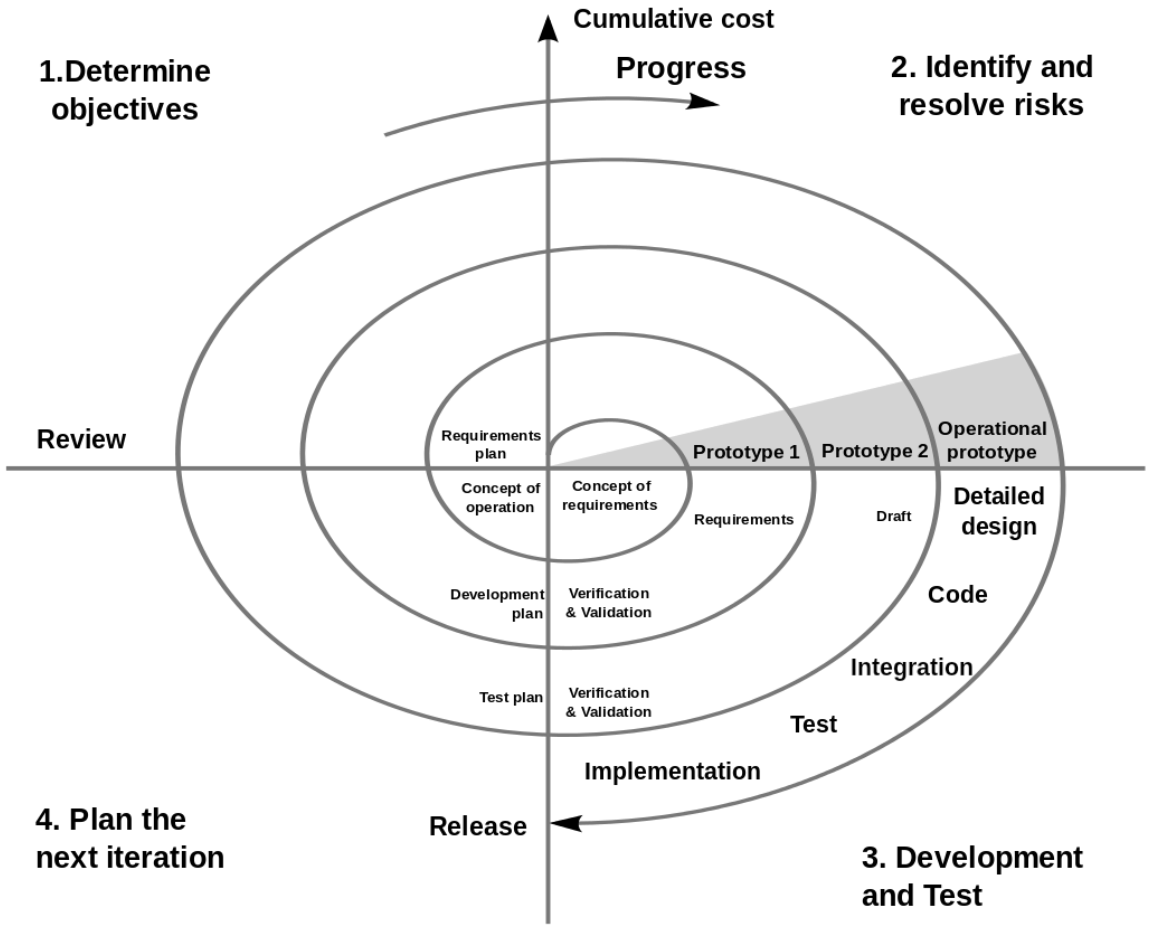
Spiral methods

In 1988, Barry Boehm published the spiral method, which is the combination of the waterfall model and rapid prototyping methodologies. It provided emphasis in a key area that many felt had been neglected by other methods such as deliberate iterative risk analysis. The basic principles include:

- Focus is on the risk assessment to minimise project risk by breaking the project into smaller segments which provides adaptability during the development process, and the opportunity to evaluate risks and weigh consideration of project continuation throughout the cycle.
- Every cycle has a progression that is the same sequence of steps, for each part of the product and for each of its levels of elaboration, from a document on the overall concept of the development to the coding of each program.
- The spiral is divided into four quadrants:
    1. Determine objectives, alternatives, and constraints of the iteration.

2. Evaluate alternatives, identify and resolve risks.
3. Develop and verify deliverables from iteration.
4. Plan the next iteration.

- Begin every cycle with an identification of stakeholders and their conditions and end each cycle with a review and commitment. (5)



Source Lines of Code

Source Lines of Code (SLOC) is frequently used to measure the size and complexity of a software project and to predict the amount of effort and time that will be required to develop a program, as well as to estimate the programming productivity once the software is produced. There are two types of SLOC measure:

1. Physical SLOC

   Physical SLOC is the count of lines in the text of the program's source code including comment lines and, in some cases, blank lines and tab spaces.

2. Logical SLOC

   Logical SLOC attempts to measure the number of executable expressions like operators and functions, but their specific definition is tied to specific computer languages.

Each approach has its own advantages and disadvantages:

- Physical SLOC us easier to measure but is very sensitive to coding style conventions and code framing.
- Logical SLOC is less sensitive to those factors yet not easy to measure.

It is important to note that SLOC is not as definite in the case of software quality and reliability as every real-life software product contains bugs and the tendency is that the larger the program, the more bugs. If we introduce a bugs/SLOC ratio, even if it remains constant, the absolute quantity of bugs grows alongside with the program's size. (6)

Number of Commits

Based on the number of commits the amount a software engineer is contributing to the code can be measured. The number can vary for each software engineer. How often one commits does not show the draw the line between a good and poor software engineer as one could commit multiple times and make simple changes. It does however give the management team a good idea on how active the software engineer is which can be used for determining promotions and support that is needed to be given.

Computational Platforms

Code review is a software quality assurance activity, which is a crucial part of software engineering. It helps to better the code quality, find defects in the program, transfer knowledge, increase the sense of mutual responsibility and to find better solutions.(7) Even looking at the performance of the software engineers is a good way to help build stronger code and reduce bugs. Code review can be very time consuming even though it is important.

However, now there are many computational platforms available for carrying out code review. Examples include Code Climate, Codebeat, Semmle, Gitcop and Gitcolony.

Code Climate

Code Climate's aim is "*Providing meaningful and actionable engineering insights for the entire engineering organization*".

The organisation was founded in 2011 by Brayn Helmkamp and Noah Davis to help solve the problem software engineers face of the assurance of software quality while being further away from the code and is currently headquartered in New York City. To solve this they built Quality, an automated code review tool, which helped over 100,000 individual software engineers worldwide save time and improve the maintainability of their codebases. Soon they realised that software quality is not the only intangible issue that managers face. Code Climate's flagship product Velocity analyses all the data from your GitHub repos and provides management with heads-up displays, real-time analytics, and custom reports to give management a clearer perspective on how the engineering team is working. Features include:

- All data is private with SSL encryption
- Activity Feeds which are pop up and show engineers when and how the code changes.
- Automated git updates so Code Climate runs when a new commit is pushed.
- Security dashboards that show a systematised catalogue of the weaknesses in the program including when they were introduced and suggestions on how to address the issue.
- GitHub integration.
- Hotspots that correlates the code quality against areas of high churn.
- Direct access to the entire development team which maximises code visibility across projects and enables team sharing. (8)

Codebeat

Codebeat actively wants to help software engineers to write clean code. "*codebeat was initially created for our own in-house use at code quest, because we needed a good tool to help us monitor the quality of our code in both web and mobile applications.*"

Codebeat is known for taking customer's feedback into consideration which is good for changing and improving. It is focused on the needs of individual software engineers and the teams too who strive to provide real business value. Their toolchain helps engineers focus on the important aspects of code review. Instead of using open source linters, they opted to generate their own algorithms from scratch. Codebeat is recognised for providing great support systems and has a well-documented API that deals with management. Languages that are currently supported include Java, Python, Ruby, Swift and Go. (9)

Gitcolony

Gitcolony aims to make the code review process more efficient. They want to deliver high quality code without reducing the team's throughput, enforce quality policies through our business rules engine and articulate QA, that ensures quality, release and development teams. Gitcolony uses gamification, allowing team members to get points for delivering high quality code and helping their team and give internal ranking based on performance to encourage engineers to surpass their capabilities as well as badges which are given to engineers based on their commitment to quality. Gitcolony helps improve code quality as well as encourage software engineers to improve and grow building better code overall. (10)

Algorithmic Approach

Machine Learning

Machine learning is a powerful 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 interaction. Because of new

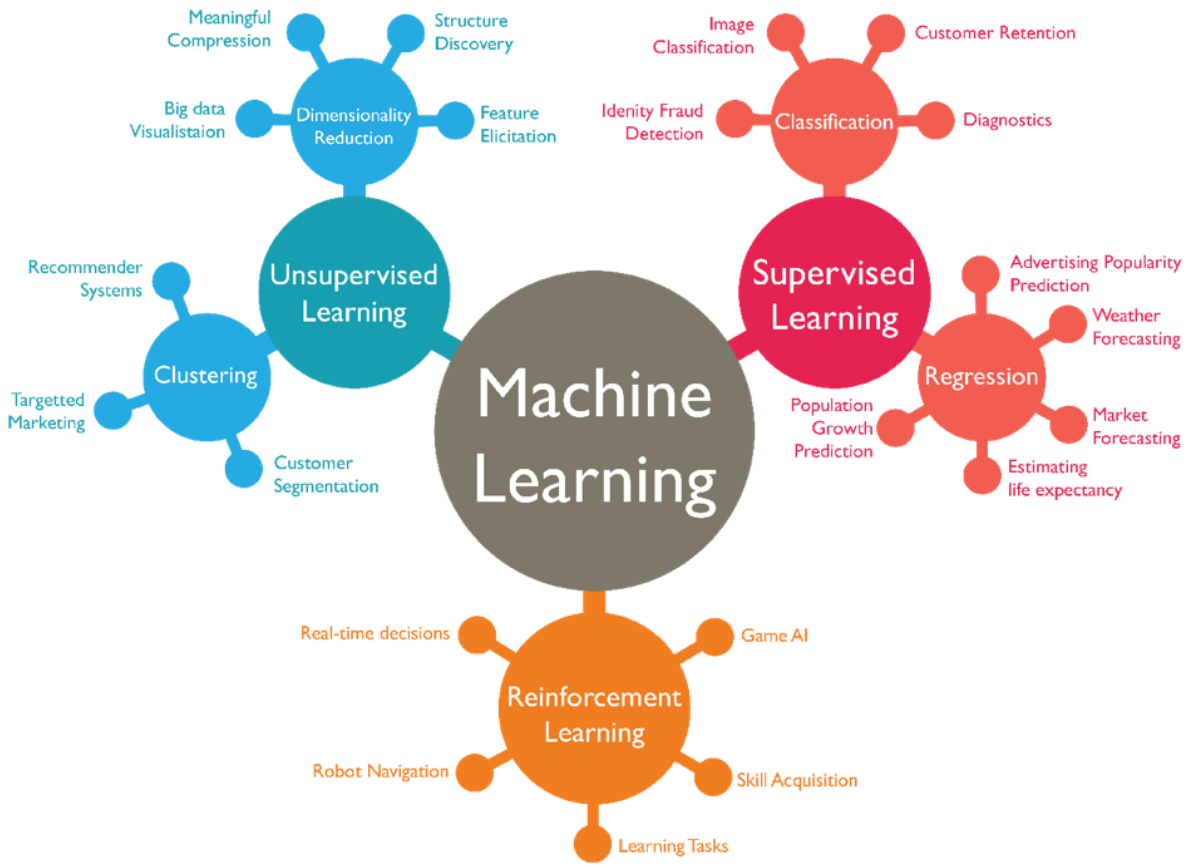
computing technologies, machine learning today is not like the past. It was born from pattern recognition and the theory that computers can learn without being programmed to perform specific tasks. They learn from previous computations to produce reliable, repeatable decisions and results. Machine learning is used for many applications including

- Semi self-driving cars.
- Online recommendation offers.
- Social media queries.
- Fraud detection.

Machine learning is important as it helps process and analyse bigger, more complex data and deliver faster and more accurate results, even on a very large scale. By building precise models, an organisation had a better chance of identifying profitable opportunities and avoid unknown risks. In order to create good machine learning systems these are required:

- Data preparation capabilities.
- Basic and advanced algorithms.
- Automation and Iterative processes.
- Scalability.
- Ensemble modelling.

Popular machine learning methods are supervised learning, unsupervised learning, semi supervised learning and reinforcement learning.



(11)

Supervised Learning

Are algorithms that are trained using labelled examples, like input where the desired output is known. Through methods like classification, regression, prediction and gradient boosting, supervised learning uses patterns to predict the values of the labels on additional unlabelled data. It is commonly used in applications where historical data predicts the likelihood of future events.

Semi Supervised Learning

Is used for the same applications as supervised learning. However, it used both labelled and unlabelled data for training. This type of learning can be used with methods such as classification, regression and prediction. Semi supervised learning is useful when the expense associated with labelling is too high to allow for fully labelled training process.

Unsupervised Learning

Is used against data that has no historic labels. The system is not told the correct answer. The algorithm must figure out what is being shown. The goal is to explore the data and find some structure within it. This type of learning works well on transactional data. Popular techniques include self-organising maps, nearest-neighbour mapping, k-means clustering and singular value decomposition.

Reinforcement Learning

Is often used for robotics, gaming and navigation. The algorithm discovers through trail and error that can lead to great results. This type of learning has three main components:

- The agent – The learner or decision maker.
- The environment – Everything the agent interacts with.
- Actions – What the agent can do.

The aim of the agent is to choose actions that maximise the expected reward over a given amount of time. (12)

Ethical Concerns

Ethical issues occur when a given decision, scenario or activity creates a conflict with society's moral principles. Ethical issues are challenging because they are difficult to deal with if no guidelines or precedents are known.

The software engineering research community is currently dealing with the ethical issue that some empirical presents. There has been an increase in software engineering research involving the use of human participants in recent years. Although the use of human participants in software engineering research seems to be more prevalent, little attention has been given to the ethical issues that come along with it. In all the other disciplines like law and medicine, the ethical issued involved in using humans has long been recognised.

The research involving human participants reported in TSE indicates that some researchers are aware of the ethical concerns raised by their research. There are instances of researchers explaining their procedures for gaining voluntary consent. However, many people aren't aware of the terms and conditions before consenting. (13)

For this reason, from a legal perspective, companies must be cautious regarding data protection. General Data Protection Regulations, GDPR, is a new set of rules designed to give EU citizens more control over their personal data. It aims to simplify the regulatory environment for business so both citizen and businesses in the European Union can fully benefit from the digital economy. As a result, companies seeking to measure their software engineering process must to do it in a legal and judicious manner or else faced a fine. (14)

In the end there is no right answer as to how to deal with the ethical concerns of software engineering. For now, ensuring that people fully understanding the terms and conditions as to what they are consenting to and the protection of private data are important in order to not face problems of increasing scale and impact.

Conclusion

This report discussed the ways in which the software engineering process can 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.

References

- https://www.techopedia.com/definition/13296/software-engineering – Definition of Software Engineering (1)

- https://link.springer.com/chapter/10.1007/978-3-540-76440-3_1 – Introduction and RoadMap: History and Challenges of Software Evolution, Tom Mens, Section 1.1 (2)

- http://w.davidfrico.com/rico09a.pdf - Use of Agile Methods in Software Engineering Education, Dr. David F. Rico and Dr.Hasan H. Sayani. (3)

- https://www.360logica.com/blog/the-importance-of-different-agile-methodologies-included-in-agile-manifesto/agile-graphic01/ – Image of Agile Methods (4)

- https://en.wikipedia.org/wiki/Spiral_model - Spiral Model article and image (5)

- https://www.viva64.com/en/t/0086/ - Source Line of Code webpage (6)

- https://en.wikipedia.org/wiki/Code_review - Code review definition (7)

- https://codeclimate.com/about/ - Code Climate webpage (8)

- https://codebeat.co/about - Codebeat webpage (9)

- https://www.gitcolony.com/features - Gitcolony webpage (10)

- https://blogs.oracle.com/ai/types-of-machine-learning-and-top-10-algorithms-everyone-should-know - Types of Machine Learning Image (11)

- https://www.sas.com/en_ie/insights/analytics/machine-learning.html - Machine Learning, SAS. (12)

- https://link.springer.com/article/10.1023/A:1011922615502 – Ethical Issues in Software Engineering Research, Tracy Hall, Valerie Flynn. (13)

- https://www.zdnet.com/article/gdpr-an-executive-guide-to-what-you-need-to-know/ - - ZDNet article on GDPR (14)