Software Engineering process: Measurement and Assessment

Objective

'The objective of this assignment is 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.'

In this report I will aim to examine and explore academic material on the subject of software engineering as well as refer to the teachings of Professor Stephen Barrett who conducts the module which I study, entitled CS3012:Software Engineering.

Introduction

In order to fully analyse ways in which the software engineering process can be measured and assessed we must first look at what software engineering is. Software engineering by definition is 'a detailed study of engineering to the design, development and maintenance of software.' (Times, n.d.) It is a process which takes the end users need into account by applying engineering principles in order to develop software. It uses a variety of programming languages in order to design, construct and test end user applications. (Techopedia, n.d.) The role of a software engineer is to not only create and develop new software but also take and modify existing software to suit their needs. A software engineer will often first think of the solution and later think of the technology needed to implement it. They can be seen as builders and designers who many a time encounter issues with their programs that they never anticipated, yet they use logical reasoning and exploration to overcome them. Often times, a software engineer's primary role is debug and fix code. As software engineer Eric Elliott once said 'Software developers write and fix bugs for a living. Sometimes some software gets made, too.' (Elliott, 2017)

The term software engineering was first introduced in the 1968 at a conference organized by NATO to manage the software crisis. This refers to the difficulties which were encountered in developing large and complex systems throughout the 1960's. The main reasoning behind this development was that costs could be reduced as well as being able to create more reliable software. (Sommerville, 2008) Although a young discipline, software engineering has undergone rapid developments and is constantly evolving.

Today, software engineering plays a pivotal role in society. It impacts all our lives on a daily basis through the various technologies we engage with. It is an exciting young, discipline

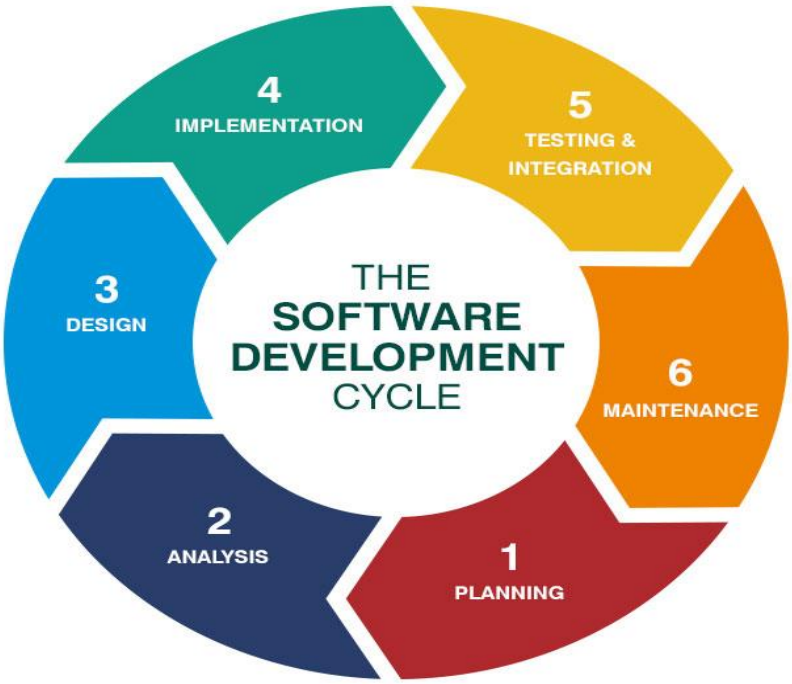
that is responsible for so much of our world today. In order to truly understand the process, I will break down this report and analyse the following.

1. The ways in which software engineering can be measured and assessed in terms of measurable data.
2. The computational platforms available to perform this work.
3. The algorithmic approaches available to deal with this data.
4. The ethics surrounding this topic.

Software engineering process

In order to be better able to analyse and discuss the above topics in this report it is fundamental that the software engineering process is understood. The software engineering process also known as the software engineering methodology, is the method used to process, plan and structure the development of information systems. (wikibooks, n.d.) Developed in the 1960's, around the same time as the introduction of name of the software engineering discipline itself, it is one of the oldest and most formal methodology for building an information system. It is the model chosen for designing and overseeing the creation of new software from the beginning to the end of the production. An important thing to note about the software engineering process is that there is a no one size fits all solution with many different methodologies available to suit a wide variety of projects.

One well known process is the systems development life cycle (SDLC) which is composed of a number of steps and clear working phases. SDLC provides a detailed plan which allows specific software to be developed, altered and enhanced. This can be broken down into six main steps which are demonstrated in the image below. (BusinessField, n.d.)



There are a number of different types of SDLC model which are all based on the same framework. Different methods suit different types of projects. The most common methods are as follows:

- Waterfalls model: This model is based on a linear sequential downwards flow.

- V-shaped model: This is an extension of the waterfalls model, however instead of moving linearly downwards the process is bent upwards following the implementation phase.

- Prototyping model: This model is designed around creating prototypes of software applications.

- Spiral Model: This method combines the advantages of top down and bottom up concepts as well as elements of both the waterfall and prototyping model.

- Iterative and incremental model: Similar to the waterfall model however it finishes with the deployment of cyclical interactions.

- Agile model: This model is based on the iterative and incremental model and evolves through collaboration between a cross functional team.

There are several other models which are not mentioned here, and these are used to deal with a variety of tasks or activities when required. When such processes are not evoked, or projects are not well managed, software engineering projects themselves can easily run over time or greatly exceed their budgets. This is why there is such a huge importance placed on software engineering processes in today's business orientated, fast paced world.

Measurable Data

"You can have data without information, but you cannot have information without data." (Daniel Keys Moran)

The first issue I wish to tackle in this report is that of measurable data. In today's world, regardless of what field of study you find yourself in, we are surrounded an enormous quantity of data. (import.io, 2018) Data is of upmost importance when we analyse the way in which we assess and measure how successful a software engineering project is.
It is important to be able measure data in software engineering for a number of reasons outlined below. (wikiversity, n.d.)

1. Planning- To know what data is required and what data is obsolete when planning a software project.

2. Organizing- To be able to decide what data is of most value to you for the project you wish to carry out.

3. Controlling- To ensure that only the data that is required to be used is being maintained.

4. Improving- To have the ability to be able to check at any point that the objective of the project is still consistent.

There are a number of metrics available to anaylse data in regard to software engineering and these metrics fit under the umbrella term known as 'software metric'. There are countless benefits to using software metrics when anaylsing a product or process. It is important to carry out these metrics throughout the course of the process in order to maintain and improve the quality of the process throughout. These methods can be used and utilised by both managers and software engineers for different purposes on development teams.

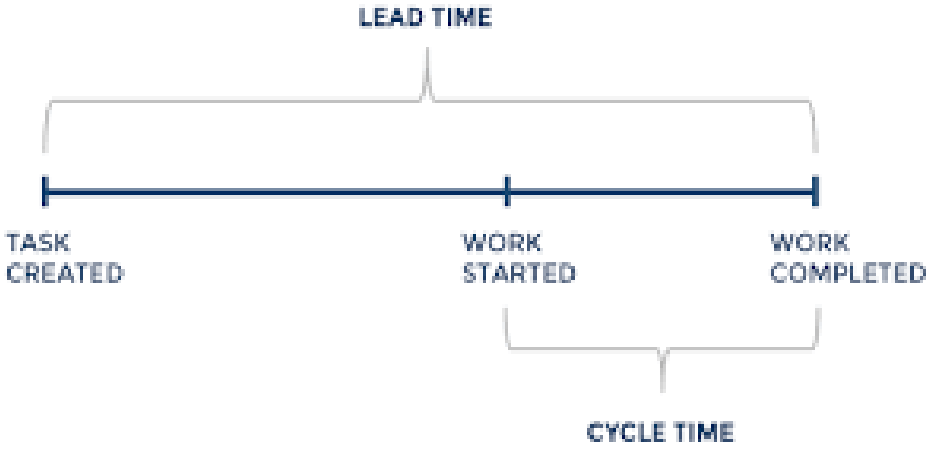
Managers can use software metrics to anaylse, prioritise and track the performance of a certain process.

Software engineers can use software metrics to communicate the status of the project as well as establish where issues lie. (stackify, 2017)

It is important to note that there is a danger surrounding software metrics and this occurs when the incorrect metrics are used, or certain metrics are used in isolation. Below is an outline of some of the most popular and effective software metrics to use.

Agile Metrics

These types of metrics are used to establish ways to enhance the process of software development. There are a number of factors which are usually taken into account and these include; lead time, cycle time, team velocity and open and closed rates. The lead time refers to the time take to come up with idea for the project while the cycle time is the time taken for developing the project. A diagram of the cycle time can be seen below.



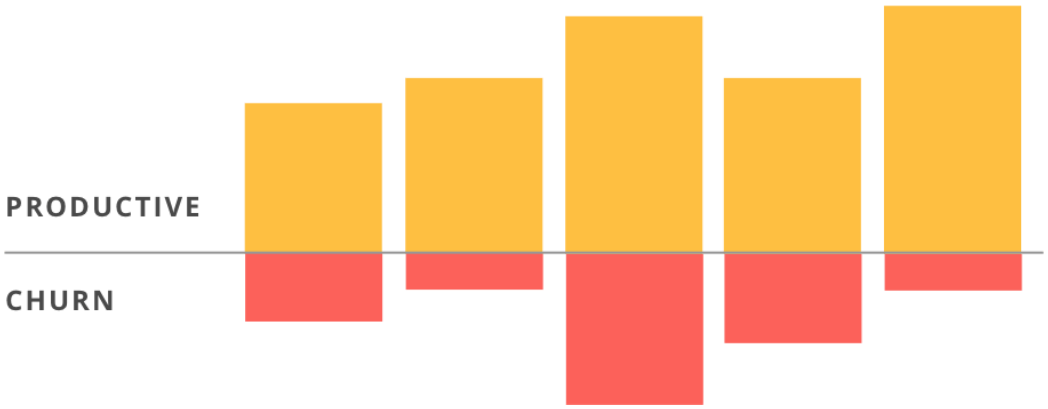
Finally the velocity of the project refers the iterations and the software engineers process on the project. (Diceus, n.d.)

Production Metrics

This type of metric takes a number of factors into consideration such as active days, task scope, productivity and code churn in order to assess the size of the project and measure productivity of the development teams.

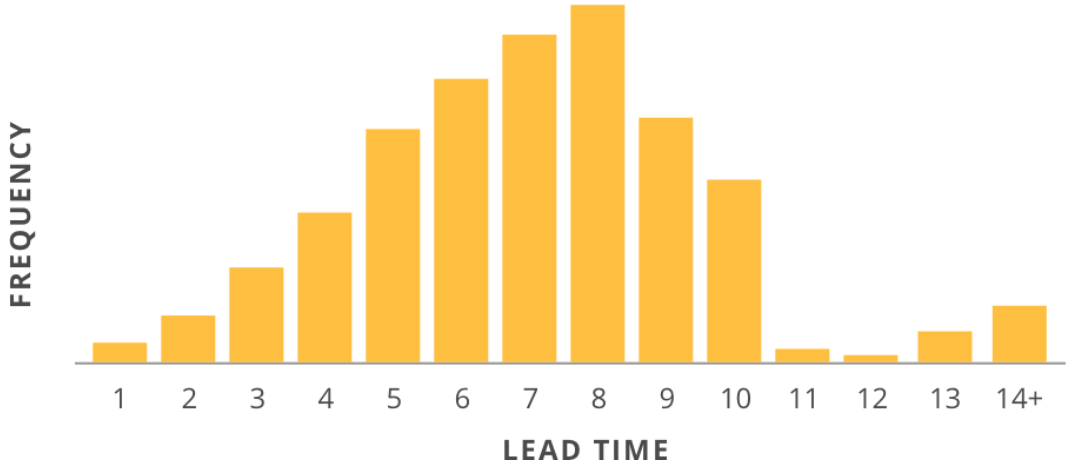
Code Churn

An example of one production metric in more detail is code churn.
Code churn is the percentage of a developers own code. It is measured in lines of code (LOC) and tracks the number of lines that were added, deleted or modified over a certain period of time. a spike in the churn signals that something is off. A graph demonstrating this is shown below. (gitprime, 2018)



Lead Time

Another production metric is lead time. This refers to the time from the beginning of the product to the time of the products development. This is a useful metric to help estimate the length of time a project will take to be completed by using past history to predict the future. A graph displaying this information can be seen below.



Number of commits

The above metics outline are best used to measure the development of a software engineering's teams process on a software engineering project, however it is also very important to measure and review how a software engineer codes and this can be achieved by recording the number of commits.

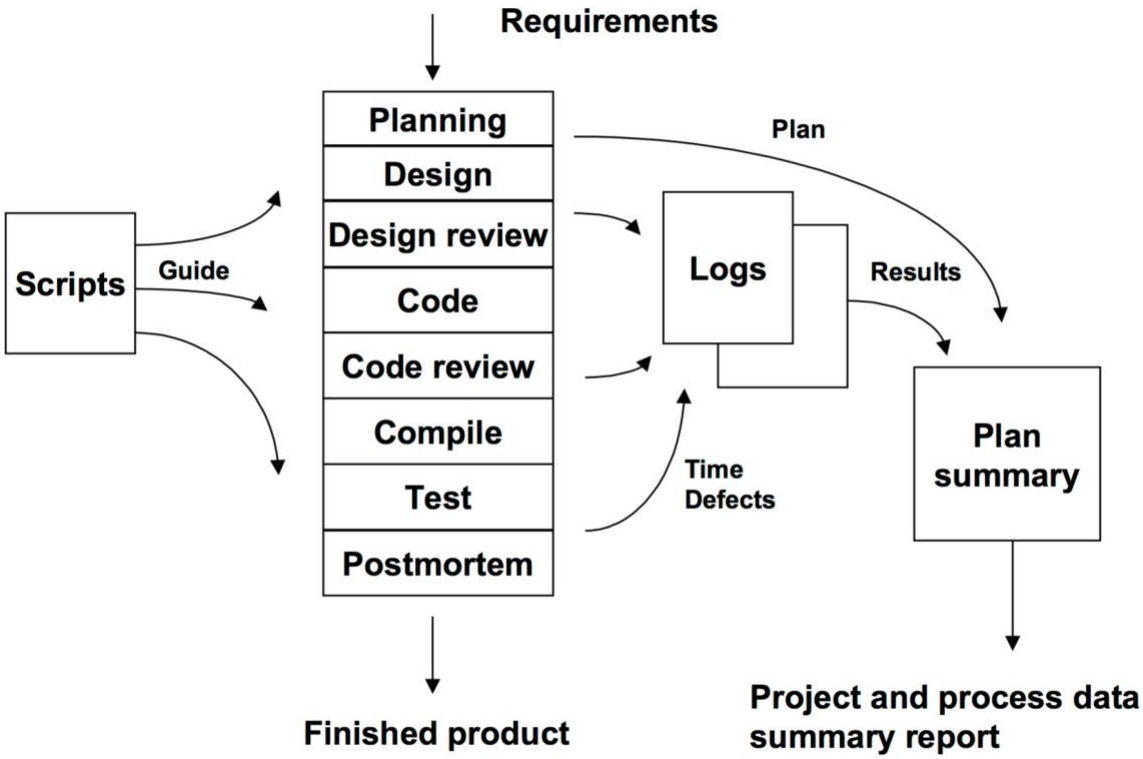
This is a key metric in measuring how an individual software engineer contributes to a software engineering process. Although the number of commits does not distinguish a good software engineer from a poor one, it merely records how many times an individual adds, removes or alters the code. This metric is effective in establishing whether a software engineer is being active or not or which individual on a certain team makes the most contributions.

Computational Platforms

In many ways the first part of the software engineering process is obtaining all measurable data, however without being able to compute information on such data, the data becomes virtually useless. In this part of the report, I hope to analyse the different platforms available for use on the data.

By definition a computational platform is known as the environment in which a piece of software is executed through user-friendly interfaces that can be used to analysed and answer questions on a given data set. (Nature, n.d.)

Until shortly after World War 2, in most industries quality strategy was predominantly based on testing. This was an expensive and time consuming method of fixing problems only after they were found. One major significant improvement to testing software quality came through Humphrey in 1995 when he developed the personal software process (PSP). This process is centralised around the individuals engineers and means that each engineer must do quality work. In order for software engineers to achieve this, Humphrey noted that each engineer must use a defined process to plan their work. The principles of this method can be seen in the PSP process flow diagram below.



(Humphrey, 2000)

After substantial research was conducted into the PSP process, it became apparent that the lengthy and manual nature of the process proved to be tedious and since then many more computational platforms were developed.

As it stands there are currently a multitude of platforms available on the market which are used to measure the software engineering process. These platforms wish to measure the performance of a software engineering process under three main headings which are highlight below. (Goldsmith, n.d)

Quality of design- This identifies what is needed in terms of capabilities, functions required and performance levels of a platform.

Quality of conformance- This is mainly centralised around how a product is produced.

Quality of performance- This deals with how a product is delivered and whether it meets the significant requirements needed of it.

A modern day computational platform is Github, which measures and analysis software engineering using a free web based hosting service. Github is an environment which allows users, namely software engineers to host respositories containing certain specifications and design mechanics as well as allowing them to add, remove and alter code for their projects. Github carries out a number of key metrics on software engineering projects such as the number of commits of code, the number of contributors to a certain project and finally the frequency of when the user accesses their project. As explained earlier in this report, this is

all invaluable in measuring and accessing the performance of a software engineer. An example of this, is demonstrated in the below heat map where we can see how active a software engineer is and hence measure their productivity.

This is just one of the many examples of a modern day computational platform which is both easy to access and easy to use.

Algorithmic approaches

Costing

When it comes to developing software engineering projects, cost estimation is a key and vital aspect. Accurate cost estimation is of upmost importance in order to ensure the project runs and is completed on time and the project doesn't exceed its budget. Algorithmic cost estimation approaches use mathematical expressions in order to most accurately estimate project costs. Algorithmic approaches use historical data to predict the future as well as taking aspects of the code such as number of functions, language and risk assessment into account.

There are a number of advantages of using algorithmic approaches when assessing software engineering projects and these are outlined below;

1. Estimations will and can be repeatable.
2. Formulas are easily modified and reformed to suit certain aspects of the input data.
3. The conclusions are objectively computed based on historical knowledge.

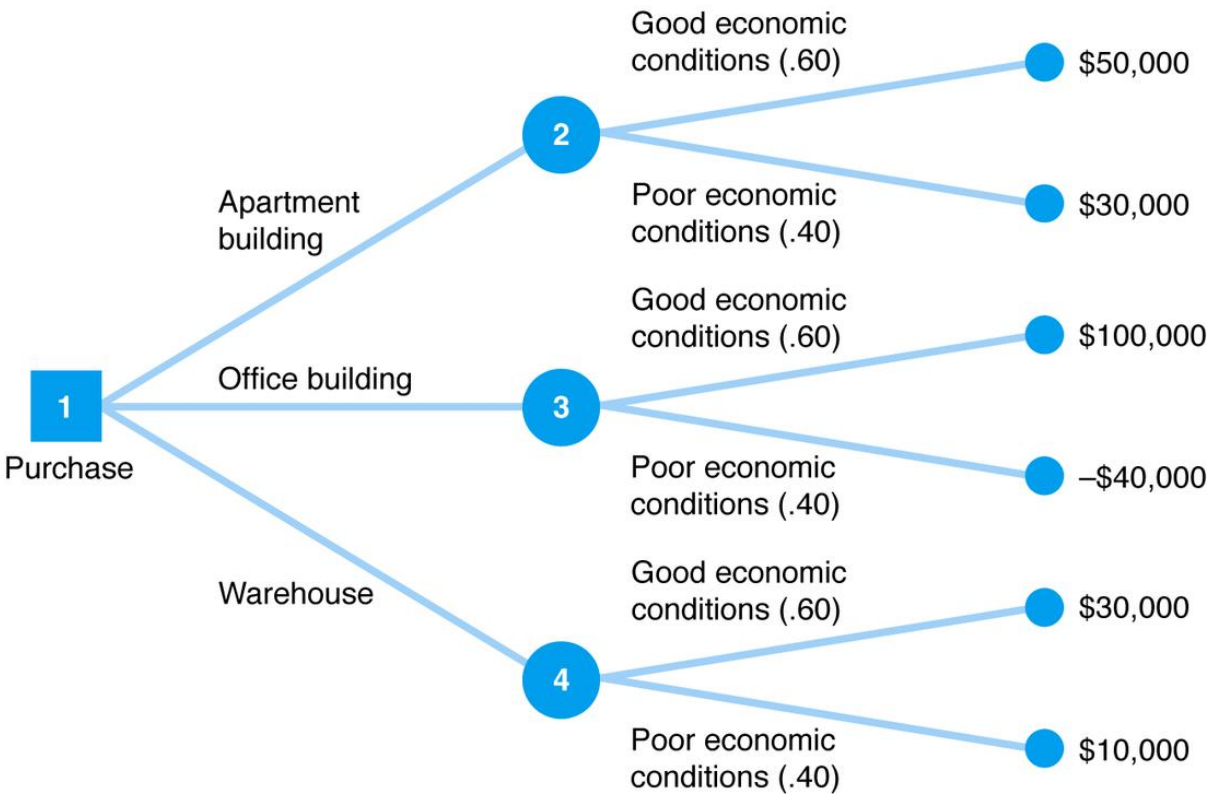
There are also some disadvantages to using algorithmic approaches such as factors not being easily quantified and certain exceptional conditions coming into play. (Potdar, 2014)

Data analysis

Algorithmic approaches can not only be used to measure the length and cost of a given software engineering project but a number of methods can also be used to gain an increased understanding surrounding the data in question. In general, when we are dealing with data used in software engineering projects we are mostly concerned with multidimensional data. That means that in order to properly analyse the data, we must break the data into two families of groups; supervised methods and unsupervised methods. (Houlding, 2018)

Supervised methods- this assumes a given structure within the data.

An example of a supervised algorithmic approach is decision tree analysis. This is a tool that use tree like structures which include aspects of chance, costs and utility to solve a problem. An example of this layout can be shown below.



(AdvancedManagementScience, n.d.)

Unsupervised methods- the structure is discovered from the data alone.

An example of an unsupervised algorithmic approach is principle component analysis. This method uses orthogonal transformations to convert a set of observations of possibly correlated values with uncorrelated values. This often makes the data set easier to analyse.

Ethics

Computers play an increasingly important role in the world around us, playing a major part in the world of commerce, government, industry and education. Software engineers have a certain responsibility as through their contributions they have a direct effect on the world around them whether that is through teaching, development or altering software. It is difficult to regulate the ethical nature of software engineers as there is no set way to go about preventing people from using code in a malicious nature however it is the responsibility of the software engineers themselves to ensure they act in a professional manner. The institute of electrical and electronic engineers have drawn up a ‘Software

Engineering Code of Ethics' in order to guide engineers to perform the correct practices. A summary of this code of ethics can be seen below.
(IEEE, 1999)

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.

It is the IEEE's and the world of software developments hope that these guidelines can be followed so that software engineering development can be carried out in a manner that is in the best interest of society.

Conclusion

To conclude, I have highlighted the key bases for measurement of the software engineering process. In this report, I have discussed how data is measured, different computational and algorithmic platforms used in regard to the software engineering process and finally the ethical nature of software engineering under clear headings. I hope this report best portrays how software engineering can be assessed and measured in an informative and easy to understand manner.

Bibliography

AdvancedManagementScience, n.d. [Online]
Available at: http://amsdecisiontreeanalysis.weebly.com/how-to-solve-problems.html
[Accessed 2018].
BusinessField, n.d. *Business Field.* [Online]
Available at: http://www.etherservices.com/single.php
[Accessed 2018].
codesimplicity, n.d. *codesimplicity.* [Online]
Available at: https://www.codesimplicity.com/post/measuring-developer-productivity/
[Accessed 2018].
Elliott, E., 2017. [Interview] (April 2017).
Goldsmith, R. F., n.d. *searchsoftwarequality..* [Online]
Available at: https://searchsoftwarequality.techtarget.com/opinion/An-expert-suggests-how-to-measure-software-quality
[Accessed 2018].
Houlding, B., 2018. s.l.:s.n.
Humphrey, W. S., 2000. [Online]
Available at:
https://resources.sei.cmu.edu/asset_files/TechnicalReport/2000_005_001_13751.pdf
IEEE, 1999. *computer.* [Online]
Available at: https://www.computer.org/web/education/code-of-ethics
import.io, 2018. *import.io.* [Online]
Available at: https://www.import.io/post/what-is-data-and-why-is-it-important/
Nature, n.d. *Nature.* [Online]
Available at: https://www.nature.com/subjects/computational-platforms-and-environments
[Accessed 2018].
Potdar, M. S. M., 2014. Literature Survey on Algorithmic Methods for Software Development Cost Estimation. *Int.J.Computer Technology & Applications,* Volume 5.
Sommerville, I., 2008. [Online]
Available at: https://ifs.host.cs.st-andrews.ac.uk/Books/SE9/Web/History/index.html
Techopedia, n.d. *techopedia.* [Online]
Available at: https://www.techopedia.com/definition/13296/software-engineering
Times, I., n.d. *The Economic Times.* [Online]
Available at: https://economictimes.indiatimes.com/definition/software-engineering
[Accessed 11 2018].
wikibooks, n.d. *Introduction to Software Engineering/Process/Methodology.* s.l.:s.n.
wikiversity, n.d. [Online]
Available at: https://en.wikiversity.org/wiki/Software_metrics_and_measurement
[Accessed 2018].