Software Engineering process: Measurement and Assesment

# 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 other 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 and structure the development of information system. (wikibooks, n.d.)Developed in the 1960’s around the same time as the introduction of software engineering it is one of the oldest and 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 to 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 (stage 4).
* 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 software projects are not well managed software engineering projects can easily run over time or exceed their budget. 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. (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 process in order to maintain and improve the quality of the process throughout. These methods can be used by both managers and software engineers themselves on development teams.

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

Software engineers can use software

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

# Software engineering process

In order to be better able to analyse and discuss the other 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 and structure the development of information system. (wikibooks, n.d.)Developed in the 1960’s around the same time as the introduction of software engineering it is one of the oldest and 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 to 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 (stage 4).
* - 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 software projects are not well managed software engineering projects can easily run over time or exceed their budget. 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](http://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. (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 process in order to maintain and improve the quality of the process throughout. These methods can be used by both managers and software engineers themselves on development teams.

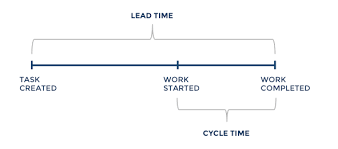
Managers can use software metrics to anaylse, prioritise and track the performance of a certain process.

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 scope 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.



Bibliography

BusinessField, n.d. *Business Field.*[Online]   
Available at: <http://www.etherservices.com/single.php>  
[Accessed 2018].

Elliott, E., 2017. [Interview] (April 2017).

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.

# Bibliography

BusinessField, n.d. *Business Field.* [Online]   
Available at: http://www.etherservices.com/single.php  
[Accessed 2018].

Elliott, E., 2017. [Interview] (April 2017).

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.