Measuring Software Engineering

Gregory Partridge- 17331009

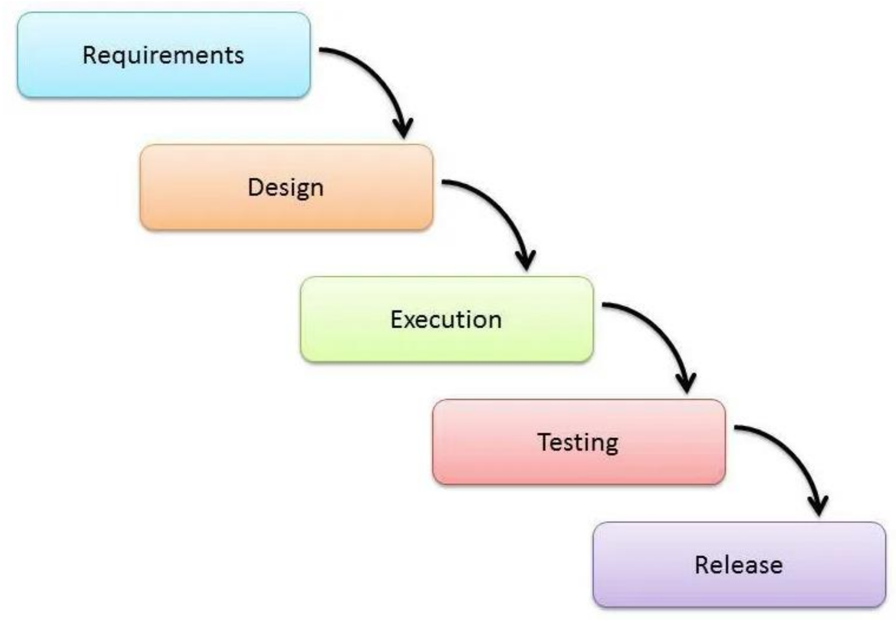
Trinity College Dublin

The University of Dublin

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 and
  - the algorithmic approaches available.
  - the ethics concerns surrounding this kind of analytics.

<u>**Software Development Process**</u>



The Software development process, or the software development cycle (SDLC) as its also called, is a series of processes that help give a understanding of the software building process. It is used in the software industry for designing, developing and testing high quality software. It can be broken down into five unique steps.

Requirements

The requirement gathering and analysis stage is where inputs from the consumer, marketing teams and surveys and the information acquired are used for quality assurance and for identifying the risks involved. The two reports are used to define the possible approaches to implement the project successfully with minimum risk.

Once that step has been implemented, the product requirements should be clearly defined and documented. This is done primarily through the making of a Software Requirement Specification(SRS) document which consist of all the requirements to be designed and developed. This should act as a guideline for the next phase.

Design

Upon receiving the SRS document, a few design approaches for the product are proposed and documented in a Design Document Specification(DDS). A design approaches is to define the architectural modules of the product. The initial design should be defined here along with all the details in the DDS.

Execution

This is the stage in which development of the project itself starts. Upon receiving the DDS document the work is divided into modules and the coding for the project starts. The developers must follow coding guidelines as well as programming tools such as compliers and interpreters are used for generating the code. The language used is chosen in respect to the software being developed. This is the main focus as it is the phase where the code is produced and is normally the longest phase of the SDLC.

Testing

Once the codes developed its passed into the next phase which is testing. This phase consists of functional testing such as unit testing, integration testing and system testing. In summary, it's where you report where the product defects, it's tracked, fixed and then retested. This is repeated until the product passes the quality standards stated in the SRS documentation.

Deployment

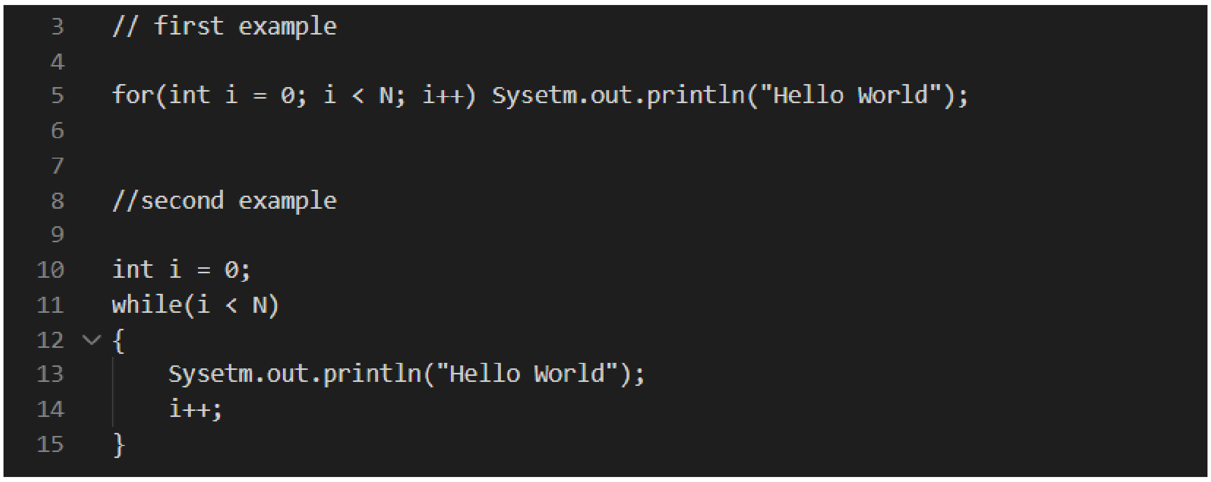
Once the product is properly tested to a high enough quality the product is then deployed to the customer for use. The product may be released in a limited segment to be tested in a real world environment. If any errors or changes are requested, they are reported they are sent back to the engineering team. Once all such changes have been finalised the product will enter its final deployment.

Measurable Data

Data is one of if not the most crucial component of the software engineering process. While it does not inherently help the software engineer in question it helps in understanding the underlying problems which may not be immediately noticeable and help arise unique solutions.

Lines of Code

This software engineering measuring quality is a more primitive type of measurement. It measures the number of lines in the program in question. There is two ways to measure LOC. The first is to only count code physical lines that end with return but many don't like it as it leads to situations where there is dead code counted as well as comments. It also brings up the question of code efficiency. The other flaw with this method in that it ignores code efficiency as seen in the below example.



```
// first example

for(int i = 0; i < N; i++) Sysetm.out.println("Hello World");


//second example

int i = 0;
while(i < N)
{
    Sysetm.out.println("Hello World");
    i++;
}
```

Both of the examples provide the same result but the first is only one line in comparison to the six line second example. To get around this flaws solutions have been suggested such as only counting only logical statements as code. From this we can show that a program could have two very different LOC counts dependent on the method used to acquire them.

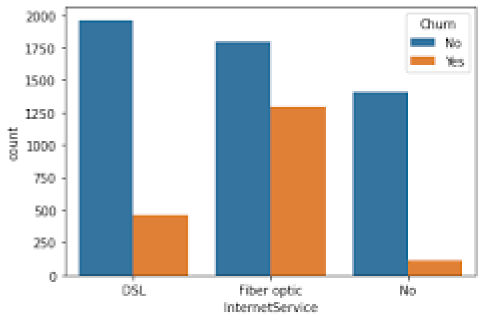
Number of Commits

This refers to the amount of changes made in the code over the duration of the product. This includes addition, removing of code, changing code or even adding comments. It is not the most effective way of measuring someone's contribution to the project but it does show how consistent they are as well as that they are focused. Large commits at a consistent level show that they are working at adding features, removing bugs and as a whole improving the quality of the program itself.

Churn

Code churn refers to a the number of lines of code that were modified, added or deleted from the code base over a set period. It allows better visualisation of the development process. If there is a spike it may indicate an area an area that needs attention. High Churn rate indicates a lack of understanding of how to identify the solution in a piece of code. On this principle it's a good indicator of post-release errors. It's a large warning if you are approaching a deadline with an increasing growing churn rate.

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6f5d3a7908>
```



Bug Fixing

Bugs are a staple of any software engineering project. There is a process in the removal and the speed of doing this is heightened by the developers knowledge of the code in question. Even though understanding of code can speed it up this process can be very unreliable to predict the speed of as some bugs can be very hard to understand never mind fixing them. A common way to speed up this process is to log out the bugs as it shows where to fix the error as well as help fix bugs that may arise in the future also.

Computational Platforms

Personal Software Process

Personal Software Process(PSP) is a software development used to help software engineers better understand and improve their performance by providing them with a disciplined personal framework for doing software work. It provides a set of methods, forms and scripts that show them how to go about the process of how to plan, measure as well as manage their own work.

HackyStat

HackyStat is an open source framework for collection, analysis, visualisation, interpretation, annotation and dissemination of software development process and product data. HackyStat users commonly attach sensors to their work and those sensers will collect and send the data collected to a HackyStat storage base. This data along with other data and can be used to generate visualisations from the raw data, abstractions or annotations. This is a step up from PSP in which the user in question had manually enter the data while here there is a level of automation added with the data being collected on its own.

Algorithmic Approaches

The industry as a whole has been on the rise since in productivity as well as efficiency since the use of artificial Intelligence and machine learning. For algorithmic approaches I will use the types of machine learning used by developers. The types of machine learning algorithms differ in their approach, data type and tasks they are intended to solve.

The three main types I will be referring to are:

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Supervised Learning

Supervised learning is the machine learning task of mapping inputs to outputs based on input output pairs. It acquires a function from training data and produces a function which may be used for mapping new examples. If we map the input with x, the output with Y we obtain the function input to output function:

$$Y = f(x)$$

This method is used as over time the algorithm learns to predict the output from the input data.

Unsupervised Learning

Unsupervised machine learning looks for previously undetected patterns in a data set which contains no pre existing labels. There is only the input data and no corresponding output data. They can be clustered into two distinct groups, clustering and association.

Clustering in which you try to group or segment data in the which to extrapolate algorithmic relationships. Association is where you want to group large groups of people under the same rules.

Reinforcement Learning

Reinforcement learning is the training of a machine to learning models in such to make a sequence of decisions. It employs a trial and error type system to come up with solution to the problem.

The core idea behind it is the programmer give the program rewards or punishments depending on how it succeeds at the task given to it. The goal of the program is to maximise its reward by combining exploration (of uncharted territory) and exploitation (of current knowledge). It starts with random trials and finishes with sophisticated skills.

Ethic Concerns

Discussing the software engineering process has brought up how to measure and how to collect data but is this truly ethical. The use of data from the workforce as shown earlier has been monumental. The question arises though where to draw the line on what is collectable data. There is the concern that data is collected that the worker did not give consent for and with the use of algorithms it is possible for that line to be too easily crossed without proper moderation. The easiest way to combat this is transparency between the employee and the employer about what data is being collected.

The data harvested should also be made sure not to leave the person identifiable and such information should be disregarded. If social media platforms are recorded it may be harder for employees to bring up dissatisfaction with the company or unionize.

Though there is all of the above there is also GDPR which helps with protecting data and privacy to all members of the EU. The main ways it impacts Software Engineers is that all data collected can not be identifiable to the given person. Consent must be ambiguous as well meaning the employer can not force you into giving up your data.

https://www.tutorialspoint.com/sdlc/sdlc_quick_guide.htm

https://melsatar.blog/2012/03/15/software-development-life-cycle-models-and-methodologies/

https://www.linkedin.com/pulse/what-software-development-life-cycle-sdlc-phases-private-limited

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

https://stackify.com/track-software-metrics/#:~:text=A%20software%20metric%20is%20a,productivity%2C%20and%20many%20other%20uses.

https://codescene.io/docs/guides/technical/code-churn.html

https://hackystat.github.io/

https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=5283

https://en.wikipedia.org/wiki/Machine_learning#Types_of_learning_algorithms

https://en.wikipedia.org/wiki/Supervised_learning

https://en.wikipedia.org/wiki/Unsupervised_learning

https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/

https://en.wikipedia.org/wiki/Reinforcement_learning

https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/

https://gdpr-info.eu/