**MEASURING SOFTWARE ENGINEERING**

Muhammad Rvail Naveed

Student No: 1732983

**Introduction**

Software Engineering can be defined as the systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software. Modern software is becoming increasingly more and more complex at an exponential rate. Since it’s inception in the late 50’s it has involved into a profession concerned with how best to maximize the quality of software and of how to create it. With technology advancing so fast, the need to understand how to make the engineering process faster and more efficient has advanced with it. Over the years there have been many discussions on how best to measure the effectiveness of the software engineering process and the productivity/competence of the developers responsible for constructing the products.

There are many metrics and methodologies that can be used to determine this. This report will discuss some of these widespread methods, their usefulness, benefits and possible ethical concerns that may arise from the collection of such data.

**Measuring and Gathering Metrics**

One of the most basic metrics that can be measured is Source lines of code (SLOC). It is used to measure the size of a program by counting the number of lines in the source. It is used to predict the effort it will take to produce a piece of software and can also be used to estimate programmer productivity once the software is complete. The basic approach is “the better the programmer, the more lines of code they will have written”. There are two main ways of measuring SLOC commonly referred to as physical SLOC and logical lines of code (LLOC). In physical SLOC the method is to count every line of code that the programmer has written, excluding comments whereas LLOC tries to determine how many statements are in the source code. Both ways can be achieved relatively easily by simple algorithms such as “wc” from GNU’s Coreutils for SLOC.

Some advantages of using SLOC as a metric for measuring the software engineering process is that it is easy to automate the process of counting the lines and many utilities to do this already exist. Another advantage is that it is an intuitive metric, it can be seen and the effect of it can be visualized. However, the bad greatly outweighs the good. As someone once said “using SLOC to measure the software progress is like using kg for measuring progress on aircraft manufacturing”. Using SLOC encourages bad practises such as copy-paste syndrome and discouragement of refactoring to make things easier. The lines of code would also depend on the experience of the developer, number of lines differs from person to person and an experienced developer may implement certain functionality in fewer lines of code than another developer of relatively less experience, even though they use the same language.

* Hours spent
* Commits?
* QA kickback rate
* Time spent on bugs
* Formal code metrics
* Dev productivity metrics
* Operational metrics
* Test metrics
* Uptime
* Flow metrics – Time, Efficiency, Distribution
* Agile methods
* Scrum methods
* Lead Time
* How metrics can help software quality

**Computational Platforms Available**

* GitHub/Gitcolony
* Jira
* Testrail

**Algorithmic Approaches**

* Halstead metrics
* AI

**Physical Approaches**

* Team ethics, work style
* Agile
* Scrum