Measuring Software Engineering

Introduction

Software engineering refers to the systematic application of engineering approaches to the development of software. The term was coined in the 1960s by Margaret Hamilton during her time working for NASA on the Apollo program. Since then, software engineering has become ever more important in the modern world, with the IT and software industries accounting for an increasingly large part of most first world economies.

Unlike most other industries, measuring production in software engineering is not a simple or straightforward task; due to the nature of the craft it is difficult to quantify the magnitude of what has been done. Despite this, however, many efforts have been made to do so, as having an idea of the volume of work done or needing to be done can be invaluable not only to software companies, but also their clients and employees. The resulting measurements may not only indicate the programming productivity of programmers and development teams, but also the maintainability of software once it has been produced. I aim to outline the various ways in which people have tried to measure software engineering and discuss the strengths and weaknesses of each.

Measurable Data

Source lines of code (SLOC)

The most obvious way to measure the 'amount' of software engineering done is to simply count the amount of text written by the software engineers. This is generally done by counting the lines of code written rather than the character or word counts. When the lines of code of a program or programs' source code are used as a metric, this is referred to as 'Source lines of code' (SLOC). SLOC is generally used to approximate the effort required to develop a program but may also be a good metric of how difficult the program will be to maintain once it has been developed.

However, there are obvious problems with this approach; not every line of code is significant, for example a lone opening or closing bracket. It also fails to account for the time and difficulty involved in writing certain sections of code compared to others. Some code may be easy to implement because it is simple in nature or is quite common in programming, while other code may require more specialist knowledge and hours bug fixing to implement successfully. Programming is more akin to problem solving than a typing test.

Also, if SLOC is used as a metric for employee performance, then this encourages more verbose code or even inefficiency in many cases. For example, a programmer may be tempted to use a 'while' loop and declare and increment a counter variable on separate

lines where a ‘for’ loop would be appropriate, or even duplicate large sections of code as opposed to writing it in a function, which would cause issues for the program’s maintainability.

SLOC where the lines in the text of a program’s sources code are counted but comment lines are excluded is referred to as *Physical SLOC.* However, this another type of SLOC called *Logical SLOC* that was developed to address many of the problems with Physical SLOC I outlined above. Logical SLOC is a measurement of the number of executable statements in a program. This has many advantages over physical SLOC as it accounts for differences in formatting and style between programmers. For example, elements such as lone brackets are not counted, and if one were to write a simple for loop with both the condition and body on the same line, this would count as one Physical SLOC but two Logical SLOC.

Logical SLOC is still far from a perfect measurement, however. It still fails to account for duplicate code and for the difficulty involved in writing, bug testing, and maintaining some blocks of statements compared to others. It also has some disadvantages over Physical SLOC in that it is more difficult to automate.

Overall, it is clear that SLOC fails to be a perfect metric of software engineering. Measuring software engineering by its most quantifiable aspect, as is done in many other industries, will never capture the full nuance of the craft. As Bill Gates said, “Measuring programming progress by lines of code is like measuring aircraft building progress by weight.” That being said, there is still merit to its simplicity and objectivity that more advanced metrics are unlikely to match.

Platforms

Algorithmic Approaches

Ethical Concerns