Anthony Tran
CS 3354.005
February 27th, 2024

Metrics Essay

Metrics are numerical measurements used to quantify a business' progress. In a business, there are many factors determining if the branch is efficient in what it is doing. Take for example, in a software development team, their efficiency could be measured in how much work they've completed. These factors can then be measured by using metrics of lines being developed, changed, or deleted. However, this can create the misconception that the higher the code lines, the more efficient the team. This would not be the case, as using metrics may make engineers more inclined to work towards the metric instead of the goal being measured. As a result, there will be a paradox where programmers would aim to complete a task in higher code lines to seem efficient, but could end up making the program more inefficient.

For a common metric, we can talk about cycle time, or how quick a feature is able to move through a lifecycle, from start to finish. In cycle time, we would have to first consider the methodology being used. A feature created in Agile can be expected to have a shorter cycle time than one created in Waterfall. Thus, one potential downfall of this is if you are comparing the metric of cycle time for Waterfall and Agile. Having a longer cycle time in Waterfall can make managers inclined to think they are being less productive than another company if they do not know the differences in methodologies. On the engineering side, engineers may strive for a shorter cycle time in order to seem more efficient. However, this could be a downside when entering the testing phase, as rushing the planning and development may lead to a higher number of bugs.

On the opposite side of the spectrum would be the metric of defects per KLOC. Here, having a lower defect number can mean the team is developing productive code that isn't prone to bugs. However, when coders are focusing on this metric, they could cause other metrics such as cycle time to increase. For example, they can spend more time on each task to ensure minimal errors, thus causing the task a longer time to finish. Furthermore, when faced with bugs, they can be unwilling to accept the bug as it is a hit to their reputation, making them less likely to accept criticism. In an environment where mistakes can lead to learning opportunities, this could create a work culture where there aren't many learning opportunities for the programmers.

In summary, metrics can be a good way to measure productivity, however, they shouldn't be taken at face value. When measuring metrics in order to find ways a team can grow, it can cause an adverse reaction instead. As a result, we shouldn't rely on numerical values, but other indicators too.