Brady Field

Beware of Counting LOC

**Important points**

“Since one of the primary purposes of counting LOC is to determine how big a system might be before we go and build it—say as an input to a project estimation process—we are presented with a dilemma. Before we build a system, we don’t have any lines of code yet, so how can we count them?” [21]

“We can determine the number of pages or lines in a book, or even weigh it, but there is no way to empirically measure its knowledge content. The same is true, unfortunately, for computer systems.” [22]

“Many estimation approaches qualify the size by assigning a ‘productivity factor’ related to the type of system. This name is misleading. People who create real-time systems are not less productive than people who create business systems even though their ‘productivity factors’ are usually lower. The factors do not address productivity, they address knowledge density.” [23]

“For estimation purposes LOC does not mean ‘line of code,’ it means ‘line of commented, test-code-written, requirements gathered, planned, designed…code.’ This is not the same as LOC.” [24]

**Disagreements**

“The ‘…not included in the product’ criterion is not relevant. The real reason we don’t count comments is scalability. We make the assumption that the comments and test code (and redundant and future code) are scalable with respect to the executable lines of code. Therefore, we don’t need to count them separately if we assume that the executable LOC contains them.” [23]

The Author here states that scalability is cited as the reason that comments and test code are not included in the LOC metric. This may be a factor, but there is another important one that was overlooked: Complexity. Complexity is a factor that is weighed heavily because it costs time. Comments, for example, are not considered complex code. The cost to produce these is very low, and therefore not as impactful as what would be considered LOC. The author dances around the issue of complexity throughout the article, such as when the measurement ‘productivity factor’ was criticized, but never actually mentions that complexity is the underlying issue of the different scores for productivity between embedded-systems and business solutions. The author does mention that the ‘knowledge density’ is higher for embedded systems, but this has more to do with an initial learning curve than it has to do with the fact that the embedded system is simply more complex. Complexity is an important part of measuring LOC that was not mentioned throughout the article.

**Questions**

I understood everything in the article.