Review of: "Software Engineering"

Timothy Lethbridge[1]

1 University of Ottawa

Potential competing interests: The author(s) declared that no potential competing interests exist.

This definition derives from the IEEE definition from many years ago. However, software engineering has evolved, so this definition has become dated and inaccurate. The key issue is that although some approaches to software engineering are 'quantifiable' (implying that measurement is central), the most important modern processes do not necessitate measurement.

Examples of such modern processes include the whole suite of agile processes (including test-driven development with automated testing, rapid update cycles including continuous integration, use of effective configuration and change management technology, user involvement, peer review, and online issue tracking), use of modern frameworks, usability evaluation with users, scalable architectures, and so on. All of these can be accurately described as systematic (conforming with this definition), and should be called software engineering only if they are undertaken in a disciplined way (also conforming with the definition). And indeed, many of these aspects *can* be tracked in a quantifiable manner (e.g. measuring test coverage percent, test case pass rate, number of issues, response time for various operations), but this sort of quantifiability is not an *essential* feature of the methods, so it is still good software engineering if relatively little quantification occurs.

The original reasons for the inclusion of the term 'quantifiable' were 1) the idea that unless one can predict how long some requirements will take to develop, one cannot determine the resources needed for the development of the software (hence the budget), and 2) unless we can measure qualities such as reliability and performance, we cannot determine whether one design is an improvement over another. The second reason for quantifiability remains, hence quantification remains useful. But the first reason has fallen by the wayside: Software engineers and other stakeholders have come to realize that one can not accurately create requirements for most types of systems a priori: Requirements naturally change, and should be allowed to do so as experience with early versions accumulates. This is core to agility, and agility is core to modern software engineering.

So I suggest that 'quantifiable' be removed from this definition.

I would also suggest the term 'evolution' be substituted to replace 'maintenance'. The latter implies that there is in initial development, and then one just fixes problems and adjusts the software to keep it running. That may be true for a bridge, a building or a mechanical entity. But properly engineered software is rarely like that; it evolves through lengthy series versions, with later versions sometimes being very different from earlier ones.