**Question:**

**Review the article by McCall, Richards & Walters (1977). Select a timelier academic article on software quality. Discuss, in 300 words, the major differences in relation to software quality between the two articles.**

McCall et al. (1977) argued that 11 categories influence software quality, while in their systematic mapping study, Yan et al. (2019) discuss ten software quality metrics which appear in more recent articles. A table is presented below, highlighting the key similarities and differences between the findings of both articles:

| Factor Influencing Software Quality | Factor mentioned by McCall et al. (1977)? | Factor mentioned by Yan et al. (2019)? |
| --- | --- | --- |
| Reliability | Yes | Yes |
| Testability | Yes | Yes |
| Efficiency | Yes | Yes |
| Portability | Yes | Yes |
| Reusability | Yes | Yes |
| Maintainability | Yes | Yes |
| Flexibility | Yes | No |
| Interoperability | Yes | No |
| Correctness | Yes | No |
| Integrity | Yes | No |
| Functionality | No | Yes |
| Understandability | No | Yes |
| Security | No | Yes |

The two lists broadly align, however, differences exist between these two articles. A mapping could be possible for certain categories even though Yan et al., do not provide definitions for the terms which they use. For instance, it could be argued that functionality in the list of Yan et al., is analogous to correctness in the list provided by McCall et al., as McCall defines Functionality by asking the question ("Does it do what I want?"), which can be seen as equivalent to functionality. Integrity in the list of McCall et al. is the same as Security in the list of Yan et al., as McCall further defines Integrity by asking the question "is it secure?".

Major categories that don't match are understandability, flexibility, and interoperability. McCall defines Flexibility by asking the question "can I change it?", and defines Interoperability by asking the question "can I interface it with another system?". Software changes over time as business requirements grow- having a flexible codebase is important so that growth can be unrestrained. In the case of interoperability, I would argue that perhaps this principle is somewhat niche- not all code necessarily interacts with other systems (such is the case with some terminal programs on Unix-based systems). McCall et al., makes no mention of understandability as a key factor, however I would argue that simple and understandable code is critical- code that is hard to understand can significantly inhibit developer productivity.

**References**

McCall, J. A., Richards, P. K., & Walters, G. F. (1977) Factors in Software Quality, Concept and Definitions of Software Quality. General Electric Company.

Yan, M., Xia, X., Zhang, X., Xu, L., Yang, D. & Li, S. (2019) Software quality assessment model: a systematic mapping study. Journal of SCIENCE CHINA Information Sciences 62(9): 1-17. DOI: <https://doi.org/10.1007/s11432-018-9608-3>