Discussion Topic: Factors which Influence Reusability

Refer to the article by Padhy et al. (2018), specifically Table 1, where the authors present a list of factors that influence the reusability of a piece of object-oriented software.

In this collaborative discussion, students must prioritise this list, presenting your argument for the priorities assigned.

My Initial Post:

As Padhy et al. (2018) mentioned, object-oriented metrics are standard and ongoing research topics in various engineering fields. Software metrics will help estimate reusable code.

Regarding prioritisation, consider that different stakeholders may have different priorities based on their unique needs and goals (Wazlawick, 2014). However, based on the overall goals of software development, I would prioritise these factors as follows:

1. Modularity is essential for software development because it allows the creation of self-contained, reusable modules that can be easily assembled to form complex systems (Lieberherr, 1988).
2. Reusability: Like modularity, reusability refers to software components' ability to be used in multiple contexts. This factor is also significant because it enables developers to reuse code rather than write new code from scratch, reducing the time and effort required for software development (Lieberherr, 1988).
3. Software understandability is essential because it ensures developers can easily comprehend the code and its functionality. As a result, it is easier to modify and extend the software (Chhabra et al. 2004).
4. Maintainability: Software maintenance is an ongoing process that includes bug fixes, feature updates, and changes as needed. It is easier to maintain software that is modular, reusable, and understandable (Erdil et al. 2003).
5. Portability: Portable software can quickly move between platforms and environments. This is significant because it allows the software to be used in various contexts (Cai et al. 2000).
6. Compatibility: Compatible software can work with other software and systems. This is important because software often needs to interact with other systems (Meyer, 1997).
7. Complexity: Complex software is difficult to understand and maintain. While complexity is often a necessary evil in software development, it should be minimised to ensure that software remains modular, reusable, understandable, and maintainable (Chong & Lee, 2015).

8. Usability: Usable software is easy for users to understand and interact with. While usability is essential, it is less critical than the other factors listed above because it primarily affects end-users rather than developers (Seffah, 2001).

References:

Padhy, N., Satapathy, S. & Singh, R.P. (2018). State-of-the-art object-oriented metrics and its reusability: a decade review. In *Smart Computing and Informatics: Proceedings of the First International Conference on SCI 2016, Volume 1* (pp. 431-441). Springer Singapore.

Wazlawick, R.S. (2014). *Object-oriented analysis and design for information systems: modelling with UML, OCL, and IFML*. Elsevier.

Lieberherr, K., Holland, I. and Riel, A. (1988). Object-oriented programming: An objective sense of style. *ACM Sigplan Notices*, *23*(11), pp.323-334.

Chhabra, J.K., Aggarwal, K.K. & Singh, Y. (2004). Measurement of object-oriented software spatial complexity. *Information and Software Technology*, *46*(10), pp.689-699.

Erdil, K., Finn, E., Keating, K., Meattle, J., Park, S. & Yoon, D. (2003). Software maintenance as part of the software life cycle. *Comp180: Software Engineering Project*, *1*, pp.1-49.

Cai, X., Lyu, M.R., Wong, K.F. and Ko, R. (2000), December. Component-based software engineering: technologies, development frameworks, and quality assurance schemes. In *Proceedings Seventh Asia-Pacific Software Engeering Conference. APSEC 2000* (pp. 372-379). IEEE.

Meyer, B. (1997). *Object-oriented software construction* (Vol. 2, pp. 331-410). Englewood Cliffs: Prentice hall.

Chong, C.Y. & Lee, S.P. (2015). Analyzing maintainability and reliability of object-oriented software using weighted complex network. *Journal of Systems and Software*, *110*, pp.28-53.

Seffah, A., Kececi, N. & Donyaee, M. (2001), December. QUIM: a framework for quantifying usability metrics in software quality models. In *Proceedings second asia-pacific conference on quality software* (pp. 311-318). IEEE.