Unit 3: Collaborative Discussion

Summary Post

My initial post argued that starting with good Requirement Analysis and a solid Architecture is the most important step for making software reusable. The feedback from my peers generally supported this idea and added some very helpful points. The main takeaway is that a "plan first" approach is the best way to build reusable software from the ground up (Frakes and Kang, 2005; Padhy et al., 2018).

Peer discussions brought up two excellent ideas. First, using "domain analysis" early on helps ensure requirements are complete and accurate. This makes it easier to spot opportunities for reuse from the very beginning and avoids problems later (Frakes and Kang, 2005). Second, it was pointed out that using existing frameworks like Django can enforce a modular design, such as the Model-View-Controller (MVC) pattern (Frakes and Terry, 1996). This is a practical way to ensure the whole team builds code consistently, which is crucial for reuse (Mehboob et al., 2021). Another interesting perspective from a different post was to prioritise Design Patterns, focusing on the practical coding solutions from the start (Gamma et al., 1994).

Putting these ideas together, it's clear that a successful reusability strategy has several parts. You need the high-level vision that comes from architecture and requirements. But you also need the right hands-on tools, like design patterns and modular programming. We all agreed that supporting factors like clear documentation and service contracts are also essential; without them, even the best components are hard to use (Buse and Weimer, 2010; Padhy et al., 2018). In the end, the best approach combines a strong plan with good coding habits to create truly reusable software.

References

Buse, R.P. and Weimer, W.R. (2010) 'Learning a metric for code readability', *IEEE Transactions on Software Engineering*, 36(4), pp. 546–558. doi:10.1109/tse.2009.70.

Frakes, W.B. and Kang, K. (2005) 'Software reuse research: Status and future', *IEEE Transactions on Software Engineering*, 31(7), pp. 529–536. doi:10.1109/tse.2005.85.

Frakes, W.B. and Terry, C. (1996) 'Software reuse: Metrics and models', *ACM Computing Surveys*, 28(2), pp.415–435. Available at: https://doi.org/10.1145/234528.234531.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1994) *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley.

Mehboob, B., Chong, C., Lee, S. and Lim, J. (2021) ‘Reusability affecting factors and software metrics for reusability: A systematic literature review’, *Software: Practice and Experience*, 51(6), pp.1259–1286. Available at: https://doi.org/10.1002/spe.2961.

Padhy, N., Satapathy, S. and Singh, R.P. (2018) 'State-of-the-Art Object-Oriented Metrics and Its Reusability: A Decade Review', in: Satapathy S., Bhateja V., Das S. (eds) *Smart Computing and Informatics*. Smart Innovation, Systems and Technologies. vol 77. Springer.