Collaborative Discussion 1 – Summary Post

In my initial post, I adopted the position that the primary causes of software-project failure, as classified by Agrawal, Walia and Anu (2024), arise from three interconnected domains: knowledge-based errors, organisational influence errors and cognitive-load errors. I emphasised that these categories operate systemically, with knowledge gaps triggering planning failures and organisational pressure amplifying human error. Peer feedback has prompted me to deepen, refine and partially re-evaluate this perspective.

Doug’s contribution, that “knowledge gaps” appear paradoxical in an era defined by information overload, encouraged me to reconsider whether the root problem is really a lack of information or instead the mismanagement of cognitive load. His references to Belabbes et al. (2023) and McBain (2025) reframed the discussion: abundance of information may overwhelm rather than empower engineers, producing the same functional outcome as insufficient knowledge. This challenged my initial assumption that knowledge-based errors primarily reflect deficits in skill or understanding.

In responding to Doug, I explored literature on cognitive overload and the role of AI in managing routine tasks. Helgesson et al. (2019) argue that cognitive load in software development environments elevates strain, which aligns with Reason’s Human-Error Model underpinning Agrawal et al.’s taxonomy. This led me to a more nuanced view: project failures may not stem from diminished human capability but from cognitive saturation intensified by organisational constraints. AI, when critically supervised, may help mitigate overload rather than contribute to a “brain drain”.

Through this exchange, my perspective has evolved from seeing “knowledge gaps” as discrete human deficits to understanding them as emergent symptoms of structural, informational and cognitive pressures. Peer dialogue refined the understanding of how these forces interact within the SDLC, reinforcing the need for integrated mitigation strategies combining organisational reform, cognitive-load management and disciplined requirements validation.

References

Agrawal, T., Walia, G.S. and Anu, V.K. (2024) ‘Development of a Software Design Error Taxonomy: A Systematic Literature Review’, SN Computer Science, 5(7), p. 467. Available at: https://doi.org/10.1007/s42979-024-02797-2.

Belabbes, M.A., Ruthven, I., Moshfeghi, Y. and Pennington, D. (2023) ‘Information overload: a concept analysis’, Journal of Documentation, 79(1), pp. 144–159. Available at: https://doi.org/10.1108/JD-06-2021-0118

Helgesson, D., Engström, E., Runeson, P. and Bjarnason, E. (2019) ‘Cognitive load drivers in large scale software development’, in 2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE). Piscataway, NJ: IEEE, pp. 91–94. Available at: https://doi.org/10.1109/CHASE.2019.00030

McBain, S. (2025) ‘Are we living in a golden age of stupidity?’, The Guardian, 18 October. Available at: https://www.theguardian.com/technology/2025/oct/18/are-we-living-in-a-golden-age-of-stupidity-technology.