Summary Post

Summary Post

by Lauren Pechey - Monday, 15 December 2025, 12:22 PM

Summary of Discussions:

Throughout this discussion on software project failures, several recurring themes have emerged regarding the factors that most frequently contribute to unsuccessful projects. Across initial and peer posts, three primary causes consistently appear: inadequate planning and unclear requirements, ineffective communication and stakeholder engagement, and insufficient risk management and testing (Agrawal, et al., 2024; Nizam, 2022). Projects with poorly defined objectives and deliverables are prone to scope creep and misalignment between technical teams and business goals, as highlighted in the NHS NPfIT and Healthcare.gov case studies (Verner & Sarwar, 2021; Scheibner et al., 2021). Similarly, knowledge-based failures often occur in smaller teams where critical expertise resides in a few individuals, illustrated by the Therac-25 incidents (Agrawal et al., 2024).

Peer responses emphasised the role of managerial competence and leadership in mitigating these risks. Ben's reflections on organisational influences and cognitive load resonated with evidence from large-scale, multi-vendor projects where miscommunication and delayed governance can amplify errors (Agrawal et al., 2024). Sergei highlighted strategic failures in projects like Windows Phone and the Berlin BER airport, demonstrating that poor planning and oversight extend beyond software development and can have systemic consequences (Sallai & Pepper, 2025; Nizam, 2022).

Collectively, the discussion reinforces that software project failure is rarely caused by a single factor. Rather, it emerges from the interplay between human, organisational, and technical elements. Effective mitigation requires robust planning, clear and continuous stakeholder engagement, rigorous risk assessment, and systematic knowledge management. Additionally, leadership must anticipate knowledge gaps, implement comprehensive testing protocols, and maintain adaptive governance frameworks to prevent minor errors from escalating.

In conclusion, understanding these failure patterns allows software development teams to design resilient projects that account for both technical and socio-organisational risks, ensuring that projects are more likely to achieve their intended outcomes.

References:

Agrawal, T., Walia, G.S., & Anu, V.K. (2024) *Development of a software design error taxonomy: A systematic literature review*. SN Computer Science, 5: 467. DOI: **https://doi.org/10.1007/s42979-024-02797-2**

Nizam, A. (2022) *Software project failure process definition*. IEEE Access, 10: 34428–34441. DOI: **https://doi.org/10.1109/ACCESS.2022.3161036**

Sallai, D., & Pepper, A. (eds.) (2025) *Navigating the 21st Century Business World: Case Studies in Management*. London: LSE Press.

Scheibner, J., Ienca, M., Sleigh, J., & Vayena, E. (2021) *Benefits, challenges and contributors to success for national eHealth systems implementation: A scoping review*. arXiv. Available at: **https://arxiv.org/abs/2106.08737**

Verner, C.M., & Sarwar, D. (2021) *Avoiding project failure and achieving project success in NHS IT system projects in the United Kingdom*. International Journal of Strategic Engineering (IJoSE), 4(1): 1–22. DOI: **https://doi.org/10.4018/IJoSE.2021010103**

Maximum rating: -

Permalink Reply