Table of Contents

[Initial Post 1](#_Toc147689378)

[Interconnected Causes 1](#_Toc147689379)

[Lack of Cooperation, Weak Task Backlog, and Insufficient Software Testing Resources 1](#_Toc147689380)

[Project Environment & Feasibility of Process Improvement 1](#_Toc147689381)

[Conclusion 1](#_Toc147689382)

[References 1](#_Toc147689383)

# Initial Post

From the paper "Perceived causes of software project failures – An analysis of their relationships" by Lehtinen et al. (2014), it's evident that understanding the intricacies of project failures requires a deep dive into not just the overt causes, but also the underlying interconnected issues that lead to these failures. Based on the study, my insights are:

## Interconnected Causes

A project doesn't fail because of a single reason; instead, there are multiple interconnected causes. On average, 130–185 causes were identified for each analysed failure in the study. Such interdependencies underscore the importance of looking at software projects holistically.

## Lack of Cooperation, Weak Task Backlog, and Insufficient Software Testing Resources

These were identified as common bridge causes. A bridge cause interconnects different process areas. Hence, these issues span across different stages and facets of the software development lifecycle.

## Project Environment & Feasibility of Process Improvement

While causes related to the project environment were frequent, they were rarely perceived as feasible targets for process improvement. This is a crucial insight because it highlights the challenge of addressing environmental factors, which might be external to the project's control.

To exemplify, consider a scenario where a software product is being developed with a weak task backlog. Due to this, there's ambiguity in the requirements. Developers might end up building features not aligned with stakeholders' needs. Coupled with a lack of cooperation among teams, the product might also lack rigorous testing. The final product, therefore, may not meet quality standards and might have functionalities that are not in sync with stakeholder requirements.

# Conclusion

Preventing software project failures requires a granular, interconnected approach. It's not just about pinpointing issues but understanding their relationships and how they impact different process areas. Improving one area might inadvertently lead to improvements in others, underscoring the need for collaboration across all stakeholders.

# References

Lehtinen, T.O.A., Mäntylä, M.V., Vanhanen, J., Itkonen, J. & Lassenius, C. (2014). Perceived causes of software project failures – An analysis of their relationships. Information and Software Technology, 56(6), 623–643.