Reducing Rework in Software Projects through Size and Complexity Metrics

Edneuci D. Audacio[1], **Vanice Dalto**[1], **Gleison M. Reis**[2]

[1] Universidade Norte do Parana- (UNOPAR)

Londrina –PR – Brazil

[2]Capgemini Brasil –São Paulo, SP – Brasil.

edneuci.audaco@cogna.com.br, vanice.dalto@cogna.com.br,
gleison.reis@capgemini.com

Abstract. *This paper proposes a model for decomposing software projects into smaller units based on the Size and Complexity Points (PTC) metric, which accounts for effort in person-hours and the required level of business rule knowledge. The primary objective is to reduce rework and enhance development efficiency. The model's evaluation was conducted through a case study in a software development company within the transportation sector, comparing a traditional management model with the proposed approach. The results indicate a significant reduction in rework, alongside improvements in project planning, delivery quality, and overall process efficiency.*

1. Introduction

The increasing reliance on software systems demands development processes that are more efficient, p redictable, a nd a ligned w ith c ost, s chedule, a nd q uality constraints [Unterkalmsteiner et al. 2012]. In this context, effort estimation emerges as one of the main challenges in project management, impacting planning and resource allocation.

Studies indicate that inaccurate estimations are directly associated with delays, increased costs, and rework [Bastos et al. 2024]. To mitigate these issues, project decomposition into smaller units has been widely adopted as a key strategy, enabling greater control, improved task detailing, and early identification of errors.

Given this scenario, this work investigates the impacts of dividing projects into smaller units, combined with a metric based on size and complexity, focusing on rework reduction in software projects.

2. Research Method

The research was conducted through a case study in a transportation-sector software development company, where client demands are recorded via Service Orders (S.O.), covering implementations, changes, and bug fixes. The study was divided into two phases:

Phase 1 (Traditional Model), where projects were treated as a single S.O.;

Phase 2 (Proposed Model), where requests were divided into smaller project units.

The analysis considered the total number of S.Os and the volume of Correction S.Os (CO), used as the primary indicator of rework.

3. Proposed Model

The model proposes the decomposition of projects into smaller units using the Size and Complexity Points (PTC) metric. Size refers to the functional volume of the request—considering screens, code, and database impact—while Complexity is associated with the required knowledge of business rules and the tools involved.

Each request is assigned a PTC score and converted into development hours using a fixed factor (Estimated Hours = Total PTC × 0.60), enabling greater accuracy in effort estimation and more effective task distribution.

Furthermore, the model adopts shorter development and testing cycles, allowing for the early identification of errors, in contrast to the traditional approach, in which testing is performed only at the end of the project.

4. Resultados

In Phase 1 (Traditional Model), three projects were evaluated, totaling 388 Service Orders and 2,263 PTC points. Of these, 178 corresponded to Correction S.Os, representing 46% rework. **In Phase 2** (Proposed Model), three projects were also evaluated, totaling 734 S.Os and 2,399 PTC points. Despite a 47% increase in the total number of S.Os, the percentage of Correction S.Os was reduced from 46% to 22%, indicating a 24% decrease in efforts associated with rework.

Additionally, it was observed that requests with equivalent effort levels, when decomposed into smaller units, generated no rework, highlighting the model's effectiveness in increasing delivery quality.

5. Conclusion

This study demonstrates that decomposing projects into smaller units, combined with a size and complexity-based metric, significantly contributes to reducing rework in software projects. The proposed approach improves delivery quality, increases process efficiency, and enables early error identification without increasing global effort.

As a limitation, we note that the study was conducted within a single organization. For future work, we recommend applying the model in different organizational contexts and comparing it with other established estimation metrics.

Referências

Bastos, C., Ávila, A. C. D. M., and Carvalho, M. A. (2024). Impacto das métricas de software no índice de manutenção: uma revisão sistemática. *Revista Científica da UNIFENAS*.

Unterkalmsteiner, M., Gorschek, T., Islam, A. M., Cheng, C. K., Permadi, R. B., and Feldt, R. (2012). Evaluation and measurement of software process improvement—a systematic literature review. *IEEE Transactions on Software Engineering*, 38(2):398–424.