AmiEs-2026 — Abstract Submission

A Trade-off-Aware Approach for Prioritized Refactoring of Dockerfiles

Délcio Cosme Monjane[1], Isabel Azevedo[1]

[1] *ISEP — Instituto Superior de Engenharia do Porto, Porto, Portugal*

1211739@isep.ipp.pt | ifp@isep.ipp.pt

Context. Docker is the de facto standard for containerization, and Dockerfiles act as executable specifications that determine how images are built. Violations of recommended practices, known as Dockerfile smells, are pervasive in open-source projects [1] and degrade security, build performance, reproducibility and maintainability [2], [3].

Problem. Linters such as Hadolint report every violation at the same level of importance, while automated repair tools such as Parfum [4] and DockerCleaner [5] fix smells in isolation. Neither indicates which smell should be addressed first when several coexist, nor makes explicit that repairing one smell may degrade a different quality attribute: pinning a dependency version, for instance, improves reproducibility while adding maintenance burden. Since developers do not perceive all smells as equally urgent [6], this absence of prioritization support leaves refactoring decisions unguided.

Objective. To design, implement and evaluate a prioritization mechanism that ranks Dockerfile repairs by their expected impact on four quality attributes, while making the trade-offs between those attributes explicit rather than hidden inside a single score.

Method. Following Design Science Research [7], a literature-derived impact matrix assigns each repair action a signed impact on security, performance, maintainability and reproducibility. A Weighted Sum Model combines these impacts with developer-defined weights to produce one priority score per detected smell. The model is implemented in Docker Prioritizer, a Python command-line tool that uses Parfum as a detection and repair oracle. It was evaluated through an automated build–repair–rebuild pipeline over 57 Dockerfiles from open-source repositories, with cold builds repeated three times per file and median values retained.

Results. Across the 24 Dockerfiles in which a smell was actually repaired, build time fell by 115 s and image size by 123 MB on average. On a held-out sample of 11 Dockerfiles not used for calibration, predicted and observed impacts agreed in direction for security (11/11), performance (10/11) and maintainability (9/11); reproducibility predictions proved least reliable.

Conclusions. Trade-off-aware prioritization of Dockerfile refactorings is feasible and yields measurable quality gains, while keeping every recommendation inspectable. The tool and the evaluation scripts are publicly available at https://github.com/delcioMonjane/dmei2.

Keywords: Dockerfile smells, automated refactoring, trade-off analysis, prioritization, multi-criteria decision making, container quality.

References

[1] Y. Wu, Y. Zhang, T. Wang, and H. Wang, "Characterizing the Occurrence of Dockerfile Smells in Open-Source Software: An Empirical Study," IEEE Access, vol. 8, pp. 34127–34139, 2020.

[2] T. Durieux, "Empirical Study of the Docker Smells Impact on the Image Size," in Proc. ICSE 2024, pp. 2568–2579.

[3] G. Rosa, S. Scalabrino, G. Robles, and R. Oliveto, "Not all Dockerfile Smells are the Same: An Empirical Evaluation of Hadolint Writing Practices by Experts," in Proc. MSR 2024, pp. 231–241.

[4] T. Durieux, "Parfum: Detection and Automatic Repair of Dockerfile Smells," arXiv:2302.01707, 2023.

[5] Q.-C. Bui, M. Laukötter, and R. Scandariato, "DockerCleaner: Automatic Repair of Security Smells in Dockerfiles," in Proc. ICSME 2023, pp. 160–170.

[6] G. Rosa, F. Zappone, S. Scalabrino, and R. Oliveto, "Fixing Dockerfile smells: an empirical study," Empirical Software Engineering, vol. 29, no. 5, 2024.

[7] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," Journal of Management Information Systems, vol. 24, no. 3, pp. 45–77, 2007.