Investigating the Link Between Programming Experience and Code Maintainability in Student Projects

Egemen Yildiz (E.yildiz-2@student.tudelft.nl)

Collaboration in Software Engineering Education

Supervisors: Dr. ir. E. Fenia Aivaloglou, MSc Merel Steenbergen

1. Introduction

In software development, maintainability is a core quality, essential for teamwork, code evolution, and long-term success. In student team projects, code quality can be highly uneven, partly due to the varying experience levels of individual contributors, which may affect the collaboration within the team.

2. Background

Previous studies have shown that programming experience affects confidence and task ownership, but the connection between experience and maintainable code is still unclear [5]. Understanding the correlation of maintainability and experience can assist lecturers when creating and providing instructions to groups in SWE projects for a more fruitful learning environment [7].

3. Research Question

What is the Relationship Between Prior Programming Experience and the Maintainability of Code in Student Software Development Projects?

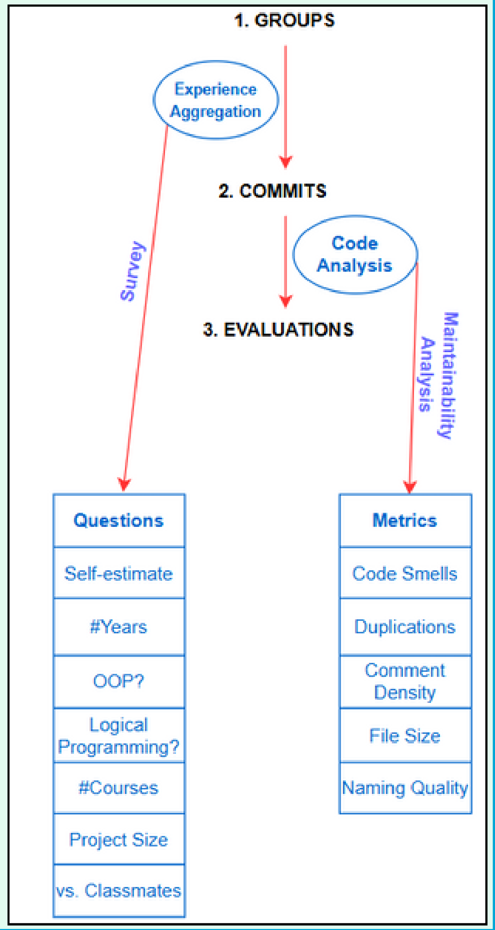
4. Methodology

20 Students, 10 week, progress on GitLab.

Data Collected:

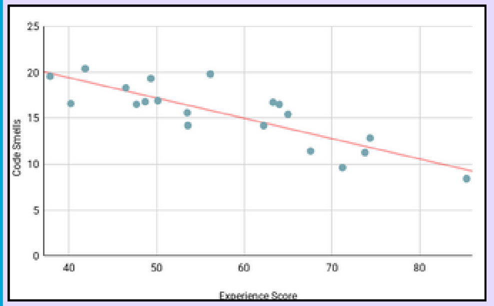
Survey, by **Feigenspan et al.** [1]

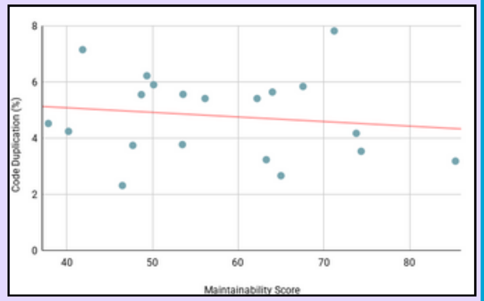
Static Code Analysis using **SonarQube**, metrics from **Ardito et al.** [2]



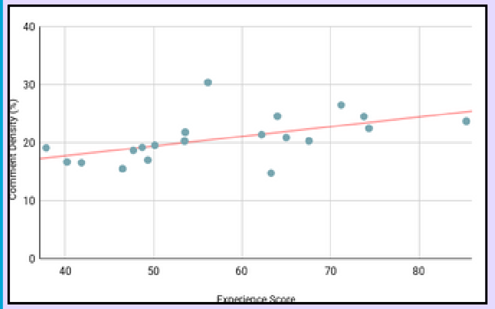
Correlations were calculated with Pearson Correlation analysis (using scipy). The normal distribution assumption was validated with the Shapiro-Wilk test.

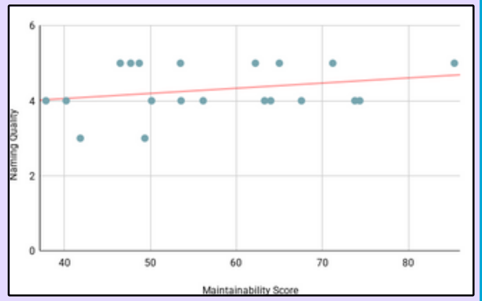
5. Results

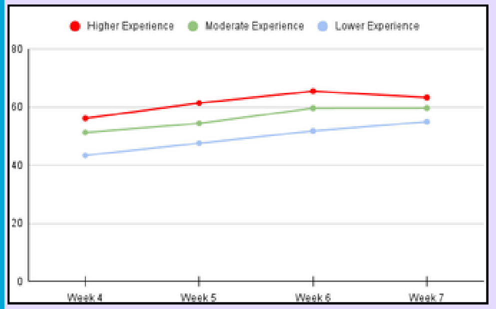


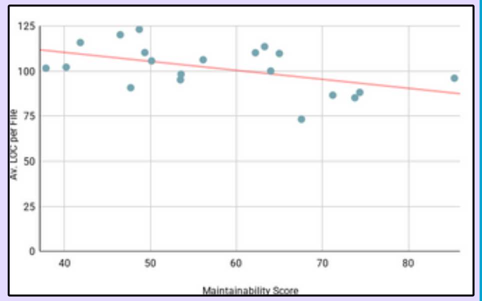


- Experience correlates with maintainability: More experienced students wrote code with fewer code smells, smaller files, and better documentation.
- Strongest metric: Code smells showed the highest correlation with experience.
- Moderate correlations: Comment density and average file size.
- Maintainability over time: All groups improved throughout the project. Lower-experience students showed the steepest improvement, narrowing the gap by Week 7.









6. Discussion

RQ1: Does experience correlate with maintainability?

✓ Consistent correlation across all five metrics
✓ Strongest for code smells, indicating experienced students better avoid structural anti-patterns
✓ Moderate links with commenting and modularity suggest exposure to norms and habits supports clarity
✓ Weaker results for naming and duplication hint at influence from rubrics, frameworks, or team dynamics

RQ2: Which metric best reflects experience?

✓ Code smells best capture internalized design skills and architectural awareness
✓ Unlike surface-level metrics, smells reflect deeper code structure and are harder to “fake”
✓ Their alignment with prior research strengthens their diagnostic value in education

RQ3: How does maintainability evolve during the project?

✓ All groups improved, but low-experience students showed steepest gains
✓ Suggests group work fosters learning from peers and exposure to better practices
✓ Reinforces the developmental value of collaborative programming environments

7. Limitations and Future Work

1) Assumed equal effort across students, though minor differences may exist.
2) Naming was scored manually, introducing subjectivity.
3) SonarQube definitions may not fully align with educational context.
4) Potential use of AI tools like ChatGPT that affects the metrics.

1) Study how team composition and collaboration influence maintainability.
2) Expand the survey to include behavioral factors like confidence or review habits.
3) Replicate study in other courses or tech stacks to test generalizability.

References:

[1] Feigenspan, J., Kästner, C., Liebig, J., Apel, S., & Hanenberg, S. (2012). Measuring Programming Experience. In Proceedings of the IEEE 20th International Conference on Program Comprehension (ICPC), pp. 73–82.
[2] Ardito, L., Coppola, R., Barbato, L., & Verga, D. (2020). A Tool-Based Perspective on Software Code Maintainability Metrics: A Systematic Literature Review. Scientific Programming, 2020, Article ID 8840389.
[3] Guaman, D., Quezada Sarmiento, P. A., Barba-Guamán, L., Cabrera, P., & Enciso, L. (2017). SonarQube as a Tool to Identify Software Metrics and Technical Debt in the Source Code Through Static Analysis. In WCSE 2017, pp. 171–175.
[4] ISO/IEC 25010:2011. Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — System and Software Quality Models.
[5] Wahler, M., Drofenik, U., & Snipes, W. (2016). Improving Code Maintainability: A Case Study on the Impact of Refactoring. In Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 493–501.
[6] Zhou, Y., Denney, E., & Fischer, B. (2021). Assessing the Students' Understanding and Their Mistakes in Code Review Checklists. arXiv preprint, arXiv:2101.04837.
[7] Garcia, J. et al. (2019). Improving Students’ Programming Quality with the Continuous Inspection Process: A Social Coding Perspective. Journal of Systems and Software, 156, 1–15.