Propositions

belonging to the thesis

Reducing Technical Debt Density:
Refactoring vs. Writing Clean New Code

by

Georgios Digkas

1. Code Technical Debt (TD) is intentionally or unintentionally created when developers introduce inefficiencies in the codebase and it can be attributed to various reasons such as heavy workload, tight delivery schedule, or developers' lack of experience.
2. The technical debt metaphor is usually associated with degrading quality trends in software systems. As systems grow in size, an increasing trend is observed for the number of issues and complexity metrics reflecting a deterioration of their TD.
3. The most frequently occurring issues regard low-level coding problems, some of which could effectively be fixed with the support of a modern IDE.
4. A small subset of all TD issue types is responsible for the largest effort on TD repayment and thus, by targeting particular violations, the development team can achieve higher benefits.
5. While TD grows in absolute numbers as software systems evolve over time, the density of technical debt (technical debt divided by lines of code) is in some cases reduced.
6. While some TD issues survive in software projects for several years, the majority of issues get fixed within one year from their introduction.
7. New code is often of better quality than the existing code base; thus, the more new code is added in each revision, the lower the risk of incurring TD interest. This entails that clean new code can be an efficient strategy for managing TD and reducing TD density.
8. Applying Quality Gates to ensure that every new revision yields fewer violations than the average, essentially leads to an improving quality trend in terms of TD and other software quality metrics.
9. The study of TD as a temporal phenomenon reveals that TD issues introduced by new code do not exhibit high fluctuations. Moreover, the number of commits and the maturity of the developers do not affect the introduction of TD.
10. Interest Generation Risk Importance (IGRI) has been proposed to capture the risk associated with incurring interest in software systems. IGRI has been shown to be capable of efficiently prioritizing TD issues to be fixed.