Table of Contents

Technical debt

Technical debt is the hidden expense of having to redo work that might have been avoided by opting for a more time-consuming, high-quality solution in the first place.

Technical debt may accrue “interest” and make improvements more difficult to execute in the same way that financial debt accumulates interest.

The entropy of software and the expense of subsequent rework rise when technical debt is not addressed.

Technical debt, like monetary debt, isn't always a terrible thing, and it's sometimes necessary to propel initiatives ahead.

There are also some experts who argue that the “technical debt” metaphor minimizes the consequences, resulting in inadequate prioritizing of the corrective effort

almbok.com_technical_debt.mp4

Methaphor - concept in software development that reflects the implied cost of additional rework caused by choosing an easy (limited) solution now instead of using a better approach that would take longer

- SonarQube
- CodeScene
- SQALE
- CLOC
- Capability Maturity Model Integration
- Squore
- Attributes
- Quality
- How-tos
- TIOBE Quality score

*Snippet from Wikipedia: **Technical debt***

> In software development, or any other IT field (e.g., Infrastructure, Networking, etc.) **technical debt** (also known as **design debt** or **code debt**) is the implied cost of future reworking required when choosing an easy but limited solution instead of a better approach that could take more time.
>
> Analogous with monetary debt, if technical debt is not repaid, it can accumulate "interest", making it harder to implement changes. Unaddressed technical debt increases software entropy and cost of further rework. Similarly to monetary debt, technical debt is not necessarily a bad thing, and sometimes (e.g. as a proof-of-concept) is required to move projects forward. On the other hand, some experts claim that the "technical debt" metaphor tends to minimize the ramifications, which results in insufficient prioritization of the necessary work to correct it.
>
> As a change is started on a codebase, there is often the need to make other coordinated

> changes in other parts of the codebase or documentation. Changes required that are not completed are considered debt, and until paid, will incur interest on top of interest, making it cumbersome to build a project. Although the term is primarily used in software development, it can also be applied to other professions.
>
> In a Dagstuhl seminar held in 2016, technical debt was defined by academic and industrial experts of the topic as follows: "In software-intensive systems, technical debt is a collection of design or implementation constructs that are expedient in the short term, but set up a technical context that can make future changes more costly or impossible. Technical debt presents an actual or contingent liability whose impact is limited to internal system qualities, primarily maintainability and evolvability."

External links:

- https://martinfowler.com/bliki/TechnicalDebt.html
- https://www.productplan.com/glossary/technical-debt/
- https://www.agilealliance.org/introduction-to-the-technical-debt-concept/
- https://www.cmswire.com/information-management/the-long-term-impacts-of-ongoing-technical-debt/