University of Groningen

Technical Debt Repayment in Practice

Tan, Jie

DOI:
10.33612/diss.195698574

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2021

Link to publication in University of Groningen/UMCG research database

Citation for published version (APA):
Tan, J. (2021). *Technical Debt Repayment in Practice*. [Thesis fully internal (DIV), University of Groningen]. University of Groningen. https://doi.org/10.33612/diss.195698574

Copyright
Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

The publication may also be distributed here under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license. More information can be found on the University of Groningen website: https://www.rug.nl/library/open-access/self-archiving-pure/taverne-amendment.

Take-down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): http://www.rug.nl/research/portal. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.

Download date: 01-02-2024

Stellingen

behorende bij het proefschrift

Technical Debt Repayment in Practice

van

Jie Tan

1. Most of the technical debt repayment effort goes into improving testing and documentation, reducing complexity, and removing duplication.

2. The majority of technical debt items in Python projects get fixed in the short-term, i.e., they are paid back within a couple of months, which is noticeably faster than in Java projects.

3. Technical debt items often co-occur with other debt items, and those pertaining to design tend to co-occur with items of similar nature.

4. A substantial amount of resolved technical debt is paid back by the same developers who introduced it; we call this self-fixed technical debt.

5. Projects that are larger, have a longer history, and accumulate more technical debt tend to have a relatively lower likelihood of observing self-fixing.

6. Practitioners are more likely to self-fix technical debt when the item is related to code-level aspects.

7. The reasons to self-fix (and introduce) technical debt are often of a non-technical nature (e.g., planning and management), although they can be combined with technical reasons (e.g., related to development process).

8. Many practitioners mention a sense of responsibility as a factor for self-fixing, and that repayment decisions are not made easily but by balancing costs and benefits, among other factors.

9. Although technical debt items take a long time to be identified and reported in issue trackers (around one year), they tend to be resolved in source code within a few days after that.