RECOMMENDING WHEN DESIGN TECHNICAL DEBT SHOULD BE SELF-ADMITTED

Computer Engineering

POLYTECHNIQUE MONTRÉAL

LE GÉNIE EN PREMIÈRE CLASSE

Cedric Noiseux

December 5, 2017

Table of Contents I

Table of Contents II

Concepts and Definitions

Technical Debts

They are not quite right code which we postpone making it right [1].

They are temporary and suboptimal solutions or workarounds.

Reasons

Quickly fix an issue

Early conception stages

Lack of comprehension, skill or experience

Technical Debts (defintion, impact) Types of Debts Self-Admitted Technical Debts (definition, exploratory study)

Low prediction (many FP) Inefficient strategies (refactoring, tracking, etc) Ineficient approaches (pattern matching, ML) Research questions (main and 3 sub-questions)

Research objective (main and 2 application scenarios) Design objectives (5 objectives)

9 OOS + Java More method-related than class-related Unbalance

Method-level Image du process (separer en differents process et expliquer) - Training and test sets - SATD matching - Features - Preprocessing - Building and applying machine learners (training and testing)

Analysis Method Within project performance Cross project performance TEDIOUS vs method level Qualitative analysis

Inspirations (Kim, Dos Santos) Description

Method-level Image du process (separer en differents process et expliquer) - Training and test sets - SATD matching - Source code -

Preprocessing/Word Embedings - Building and applying machine learners (training and testing)
Research questions (main and 3 sub-questions) 9 OOS + Java More method-related than class-related Unbalance
Source code comments only Source code with comments Source code without comments Source code partially with comments
Construct Internal Conclusion Reliability External
Number of metrics/warning Number of LOC Process inherent errors Default configurations (no real optimizations) Generalization of results Only in Java Performance evaluation can be better for CNN TEDIOUS approach CNN approach Dataset TEDIOUS results CNN results Applicability scenarios - Recommendation system - Complement to smell detectors Improvements - Optimization - Pattern matching process - More examples to train - More positive examples - More metrics/warnings and source code - Other TD types - Extend to other languages and domains

Bibliography

TEX, LATEX, and Beamer

[1] W Cunningham. *The wycash portfolio management system*. dans Addendum to the Proceedings on Object-oriented Programming Systems, Languages, and Applications (Addendum), série OOPSLA ’92, New York, NY, USA: ACM, 1992, pp 29–30, DOI: 10.1145/157709.157715. En ligne: http://doi.acm.org/10.1145/157709.157715.