lab-soft
software engineering laboratory

Detecting and Characterizing Self-Admitted Technical Debt:
A Systematic Literature Review

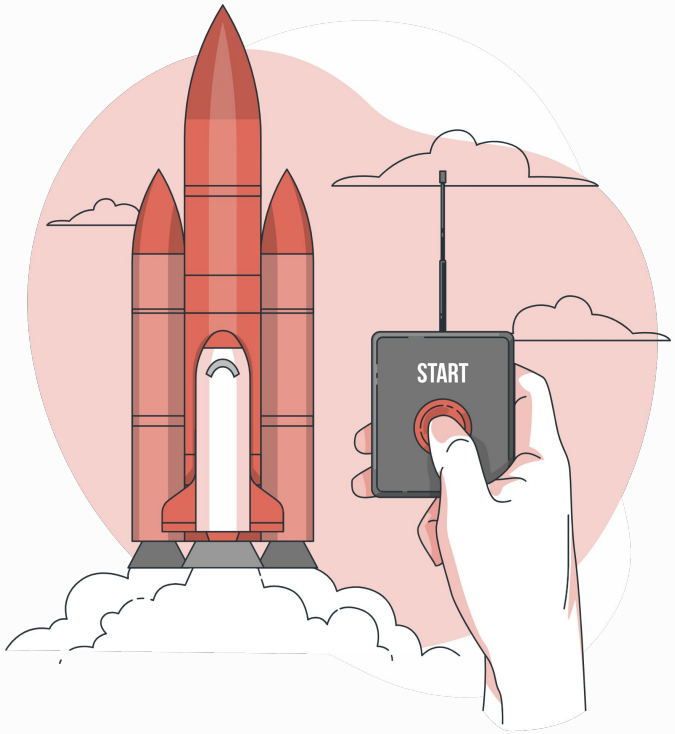
Altino Alves Júnior

Abril 13rd, 2024

lab-soft
software engineering laboratory

A research by:

- Altino Alves Júnior
- Eduardo Figueiredo
- Cleiton Tavares



01

Introduction

What’s Technical Debt and SATD?

Technical Debt

Ward Cunningham (1992) first introduced the concept of considering the “not-quite-right code” as a form of debt. Technical debt is a metaphor introduced to describe the situation where long-term code quality is traded for short-term goals.

- However, **technical debt <u>is not always visible</u>**.

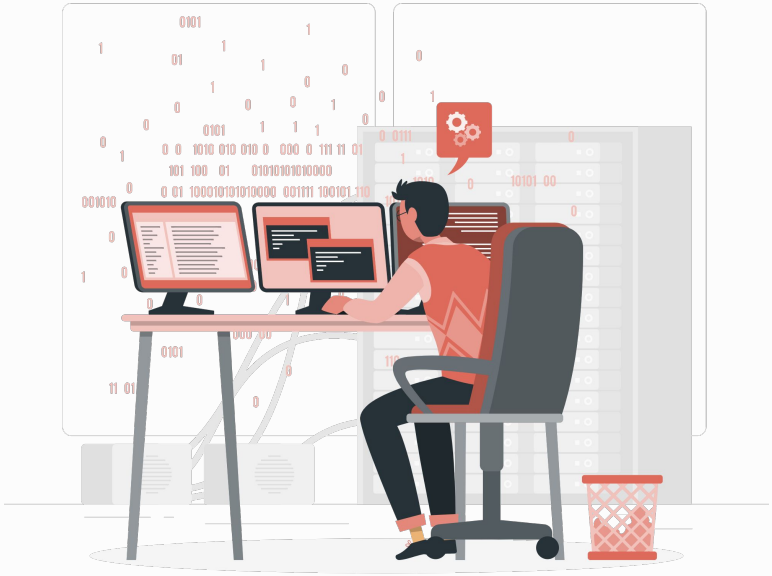
Cunningham, Ward. "The WyCash portfolio management system." ACM Sigplan Oops Messenger 4.2 (1992): 29-30.

Self-Admitted Technical Debt

Potdar and Shihab (2014) proposed the concept of self-admitted technical debt **(SATD)**, which considers debt that is intentionally introduced.

- i.e., Code that's either incomplete, defective, temporary or simply sub-optimal.

Developers document this using code comments or system messages.



A. Potdar and E. Shihab, "An Exploratory Study on Self-Admitted Technical Debt," 2014 IEEE International Conference on Software Maintenance and Evolution, Victoria, BC, Canada, 2014, pp. 91-100, doi: 10.1109/ICSME.2014.31.

Impact

Impact on Software Quality

S Wehaibi, E Shihab, L Guerrouj examine the relation between self-admitted technical debt and software quality by investigating whether:

- Files with self-admitted technical debt have more defects compared to files without self-admitted technical debt;
- Self-admitted technical debt changes introduce future defects;
- Self-admitted technical debt-related changes tend to be more difficult.

S. Wehaibi, E. Shihab and L. Guerrouj, "Examining the Impact of Self-Admitted Technical Debt on Software Quality," 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Osaka, Japan, 2016, pp. 179-188, doi: 10.1109/SANER.2016.72.

02

Protocol

Systematic Literature Review Protocol

Systematic Literature Review

Concept

A Systematic Literature Review (SLR) is the way to identify, evaluate, and interpret all available research relevant to a **singular research question**, topic area, and/or interest. This involves executing a series of meticulously planned phases:

I. planning;
II. conducting;
III. reporting.

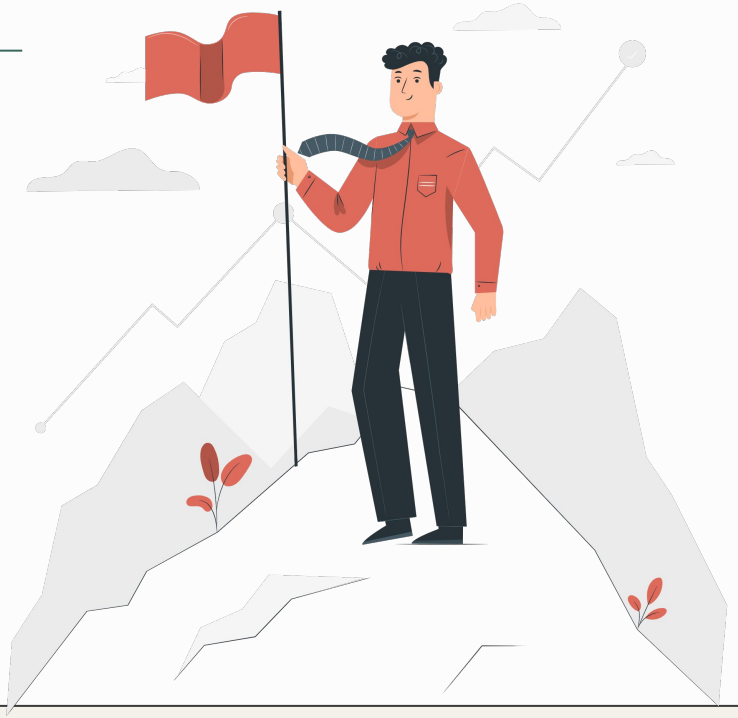
Ana Melo, Roberta Fagundes, Valentina Lenarduzzi, and Wylliams Barbosa Santos. 2022. Identification and measurement of requirements technical debt in software development: a systematic literature review. Journal of Systems and Software, 194, 111483.

Systematic Literature Review

Goal

The main goal is to explore and document all SATD detection strategies reported and used in the literature.

- Understand how different ways to detect and **classify** whether a code comment is SATD, as well as, if applicable, **relevant characteristics** thereof.



Systematic Literature Review

Research Questions

For this, study we have defined three general Research Questions (RQs), described as follows:

- **RQ1.** What are the SATD detection strategies proposed in literature?
- **RQ2.** How are SATD detection strategies evaluated?
- **RQ2.** How can the strategy be applied to SATD detection?

Systematic Literature Review

Research String

As mentioned, the goal is to find the largest set of available for SATD strategy detection. After a pilot study conducted with the aim of testing different search strings, finally, the search string was consolidated as follows:

(("SATD" **OR** "Self-Admitted Technical Debt"**)**
AND
("detect" **OR** "mining" **OR** "identify"**))**

Systematic Literature Review

Eletronic Data Sources

The goal of this study is to find largest set of available for SATD strategy detection. Initially, a pilot study was conducted with the aim of testing different search strings, thus applying different terms to all the bases mentioned. Therefore, the result was evaluated in each database in order to identify which of the strings reached the largest possible number of studies in the literature associated with the goal. Finally, the search string was consolidated.

Systematic Literature Review

Electronic Data Sources

	URL	Amount
ACM Digital Library	http://dl.acm.org/	39
Engineering Village	https://www.engineeringvillage.com	104
IEEE Xplore	http://ieeexplore.ieee.org/	91
Scopus	http://scopus.com/	73
Springer	http://link.springer.com/	900
Web Of Science	http://apps.webofknowledge.com/	60

Systematic Literature Review

Study Selection Process

Inclusion and Exclusion Criterias

Inclusion	Exclusion
Written in English	< 5 pages
Published in conferences, journals, workshops	Thesis, dissertations, tutorials, courses and magazines issues
Available in electronic format	

Systematic Literature Review

Study Selection Process

Selection Procedure

03

Detection

SATD detection methods and strategies

Detection

In the life cycle of **SATD**, debt instances are first introduced by developers into the source code. Thus naturally, the first step to study this phenomenon is to identify it.

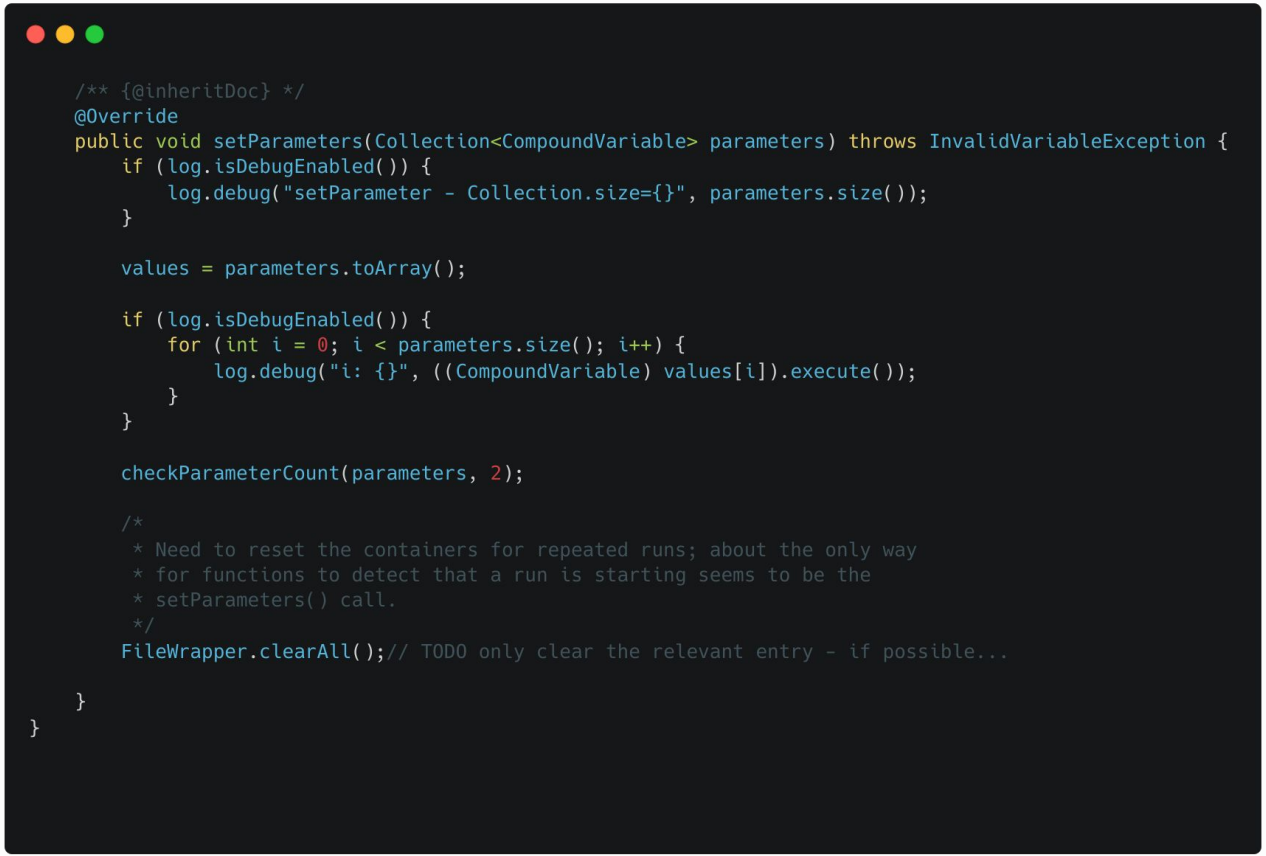
“TODO: - This method is too complex, lets break it up”

from ArgoUml

“Hack to allow entire URL to be provided in host field”

from JMeter

A. Potdar and E. Shihab, "An Exploratory Study on Self-Admitted Technical Debt," 2014 IEEE International Conference on Software Maintenance and Evolution, Victoria, BC, Canada, 2014, pp. 91-100, doi: 10.1109/ICSME.2014.31.
E. d. S. Maldonado, E. Shihab and N. Tsantalis, "Using Natural Language Processing to Automatically Detect Self-Admitted Technical Debt," in IEEE Transactions on Software Engineering, vol. 43, no. 11, pp. 1044-1062, 1 Nov. 2017, doi: 10.1109/TSE.2017.2654244.



```java
    /** {@inheritDoc} */
    @Override
    public void setParameters(Collection<CompoundVariable> parameters) throws InvalidVariableException {
        if (log.isDebugEnabled()) {
            log.debug("setParameter - Collection.size={}", parameters.size());
        }

        values = parameters.toArray();

        if (log.isDebugEnabled()) {
            for (int i = 0; i < parameters.size(); i++) {
                log.debug("i: {}", ((CompoundVariable) values[i]).execute());
            }
        }

        checkParameterCount(parameters, 2);

        /*
         * Need to reset the containers for repeated runs; about the only way
         * for functions to detect that a run is starting seems to be the
         * setParameters() call.
         */
        FileWrapper.clearAll();// TODO only clear the relevant entry - if possible...

    }
}
```

from JMeter | CSVRead.java File - Edited with Carbon

Detection Strategies

Pattern-based approaches

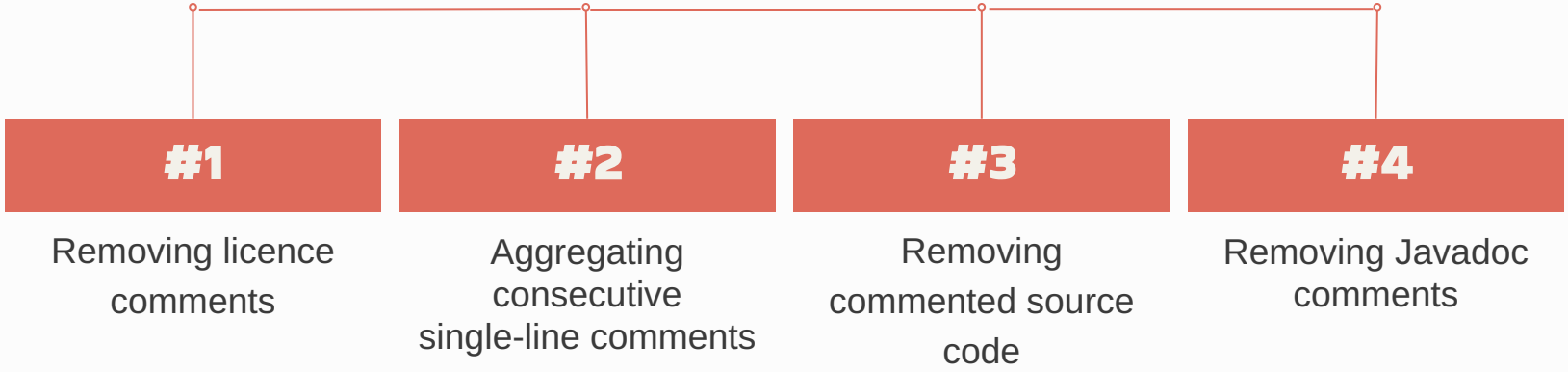
One of the main study about, Potdar and Shihab (2014) finded **62 patterns** and made them publicly available to enable further research.

- Some examples are: hack, fixme, is problematic, probably a bug, hope everything will work, etc.

A. Potdar and E. Shihab, "An Exploratory Study on Self-Admitted Technical Debt," 2014 IEEE International Conference on Software Maintenance and Evolution, Victoria, BC, Canada, 2014, pp. 91-100, doi: 10.1109/ICSME.2014.31.

Detection Strategies

SATD filtering heuristics



A. Potdar and E. Shihab, "An Exploratory Study on Self-Admitted Technical Debt," 2014 IEEE International Conference on Software Maintenance and Evolution, Victoria, BC, Canada, 2014, pp. 91-100, doi: 10.1109/ICSME.2014.31.
E. d. S. Maldonado, E. Shihab and N. Tsantalis, "Using Natural Language Processing to Automatically Detect Self-Admitted Technical Debt," in IEEE Transactions on Software Engineering, vol. 43, no. 11, pp. 1044-1062, 1 Nov. 2017, doi: 10.1109/TSE.2017.2654244.

Detection Strategies

Pattern-based approaches

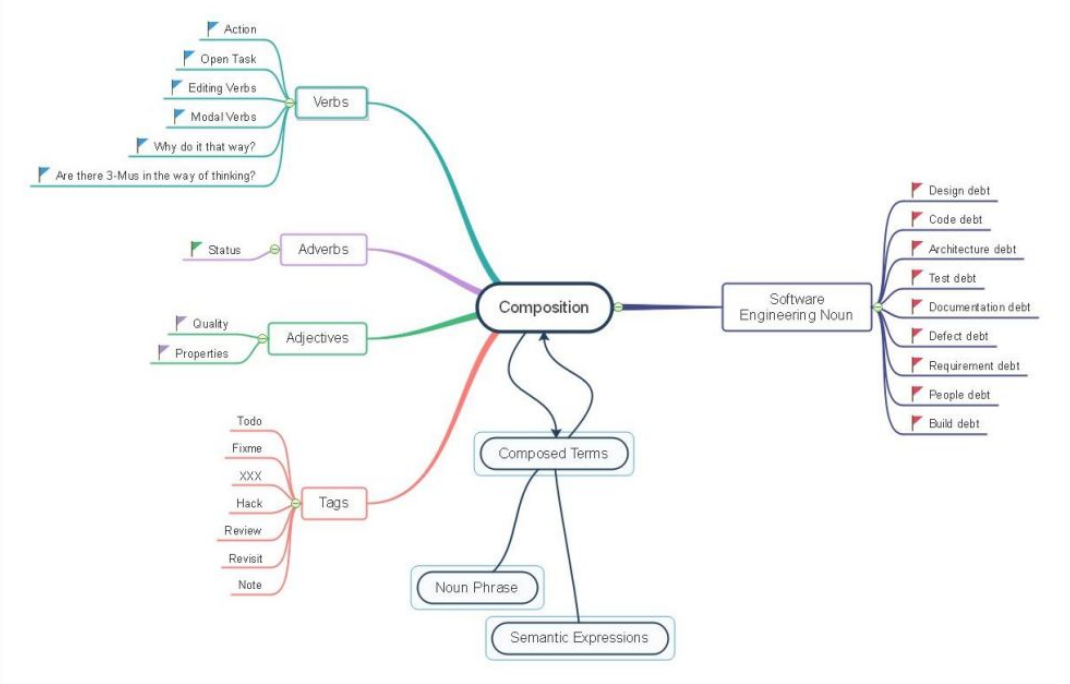
Contextualized Vocabulary Model

An alternative and extension to the pattern-based detection approach was later proposed by de Freitas Farias et al. (2015), who introduced CVM-TD for Identifying TD of different types in source code comments.

This model relies on identifying word classes, namely: **nouns**, **verbs**, **adverbs**, and **adjectives** that are related to Software Engineering terms and code tags.

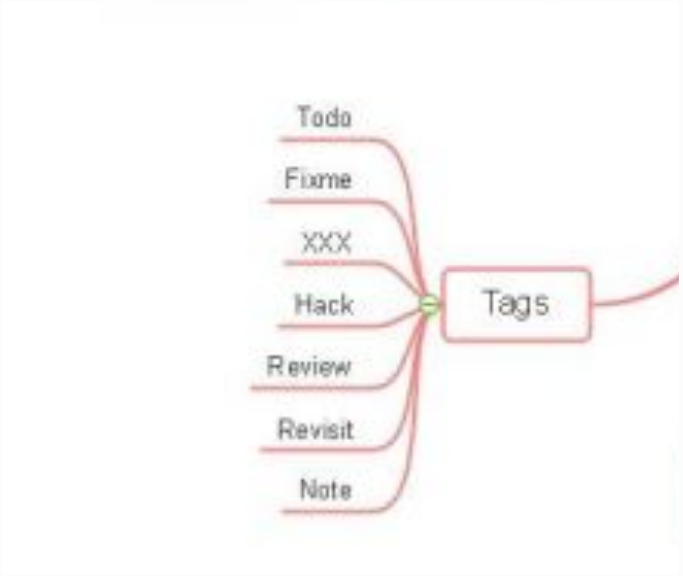
M. A. de Freitas Farias, M. G. de Mendonça Neto, A. B. d. Silva and R. O. Spínola, "A Contextualized Vocabulary Model for identifying technical debt on code comments," 2015 IEEE 7th International Workshop on Managing Technical Debt (MTD), Bremen, Germany, 2015, pp. 25-32, doi: 10.1109/MTD.2015.7332621.

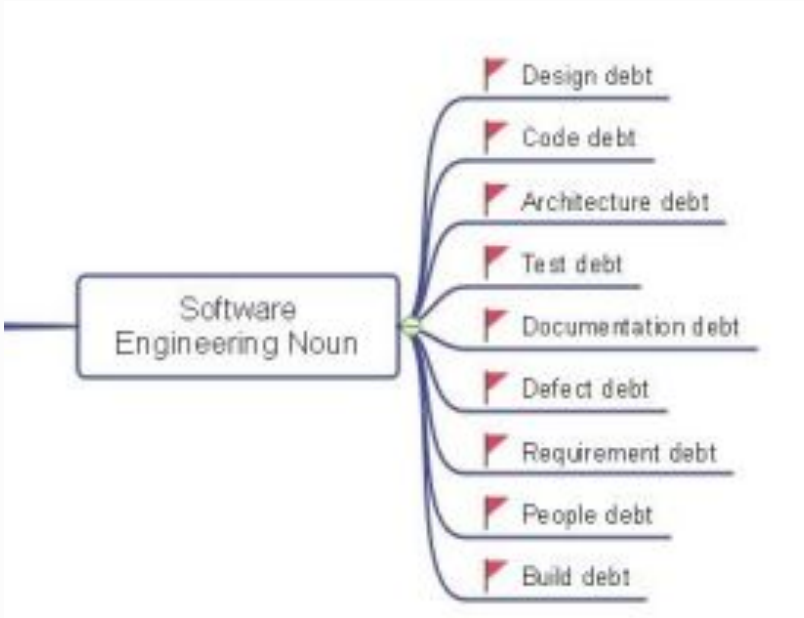
Contextualized Vocabulary Model



M. A. de Freitas Farias, M. G. de Mendonça Neto, A. B. d. Silva and R. O. Spínola, "A Contextualized Vocabulary Model for identifying technical debt on code comments," 2015 IEEE 7th International Workshop on Managing Technical Debt (MTD), Bremen, Germany, 2015, pp. 25-32, doi: 10.1109/MTD.2015.7332621.

Contextualized Vocabulary Model





M. A. de Freitas Farias, M. G. de Mendonça Neto, A. B. d. Silva and R. O. Spínola, "A Contextualized Vocabulary Model for identifying technical debt on code comments," 2015 IEEE 7th International Workshop on Managing Technical Debt (MTD), Bremen, Germany, 2015, pp. 25-32, doi: 10.1109/MTD.2015.7332621.

Text Mining

The process of exploring, analyzing and transforming large amounts of unstructured text data aided by software that can identify **concepts**, **patterns**, **topics**, **keywords** and other interesting **attributes** in the data.

Doing so typically involves the use of **natural language processing** (NLP) technology, which applies computational linguistics principles to parse and interpret data sets.

Detection Strategies

Machine learning approaches

Natural Processing Language (NLP)

Refers to the branch of computer science, artificial intelligence – concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

NLP combines computational linguistics with **statistical**, **machine learning** and **deep learning models**.

Detection Strategies

Machine learning approaches

Huang et al. (2020) proposed an approach to **automatically** detect SATD using text mining and a composite classifier, named **Ensemble text mining** approach. It's root concept is to determine if a comment indicates SATD or not based on training comments from different software projects.

- This approach preprocesses comments by **tokenizing**, **removing stop-words** and **stemming their descriptions** to obtain textual features.

Zhongxin Liu, Qiao Huang, Xin Xia, Emad Shihab, David Lo, and Shanping L. SATD detector: a text-mining-based self-admitted technical debt detection tool. In Proceedings of International Conference on Software Engineering: Companion Proceeedings (ICSE). Association for Computing Machinery, New York, NY, USA, 9–12. https://doi.org/10.1145/3183440.3183478

04

Others

Comprehension and Future Work

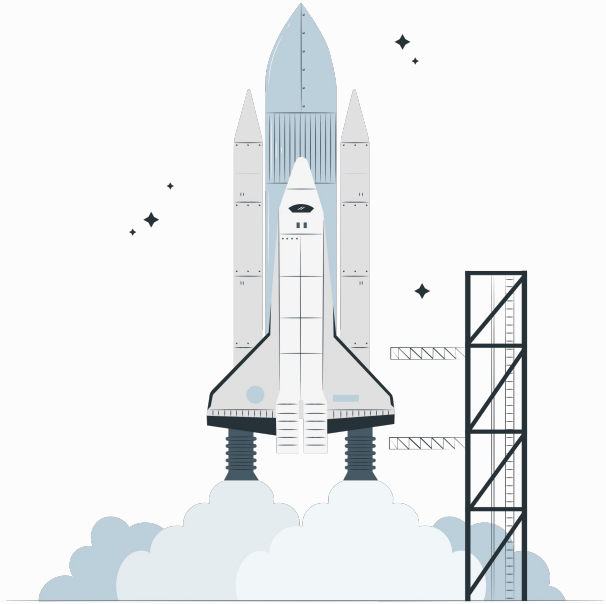
Comprehension

Types of SATD

	Example	Project
Design Debt	“/*TODO: really should be a separate class */”	ArgoUml
Defect Debt	“Bug in the above method”	Apache JMeter
Requirement Debt	“//TODO no methods yet for getClassname”	Apache Ant
Documentation Debt	“**FIXME** This function needs documentation”	Columba
Test Debt	“//TODO enable some proper tests!!”	Apache JMeter

M. A. de Freitas Farias, M. G. de Mendonça Neto, A. B. d. Silva and R. O. Spínola, “A Contextualized Vocabulary Model for identifying technical debt on code comments,” 2015 IEEE 7th International Workshop on Managing Technical Debt (MTD), Bremen, Germany, 2015, pp. 25-32, doi: 10.1109/MTD.2015.7332621.

Conclusion + Future Work



- Compilation of information to answer the RQs;
- Writing the results section, including discussions, graphics and tables;
- Writing of other sections;
- Text review.

Thanks!

Do you have any questions?

altino@ufmg.br