AnaConDebt: A Tool to Assess and Track Technical Debt

Antonio Martini
University of Oslo
Oslo, Norway
antonio.martini@ifi.uio.no

ABSTRACT

It is challenging to assess and manage Technical Debt. Technical Debt is avoided or refactored if the long-term benefits, such as preventing extra-costs, exceed the cost of repaying the debt.

Some tools have been recently proposed for the identification of Technical Debt, but most of them do not help in assessing the cost-benefits of repaying the Debt. Besides, it is challenging to track, visualize and plan Technical Debt refactoring systematically. Although practitioners might use simple tracking tools, calculating and communicating Technical Debt is currently not supported.

Based on the results of previous research, combined with several practical experiences in collaboration with large software companies, we have developed and evaluated a lightweight tool, AnaConDebt, to track and assess Technical Debt.

KEYWORDS

Technical Debt; Tool; Software Management

ACM Reference format:
Antonio Martini. 2018. AnaConDebt: A Tool to Assess and Track Technical Debt. In *Proceedings of TechDebt '18: International Conference on Technical Debt , Gothenburg, Sweden, May 27–28, 2018 (TechDebt '18),* 2 pages.
https://doi.org/10.1145/3194164.3194185

1 INTRODUCTION

Technical Debt is regarded as sub-optimal solutions that provide benefits in the short term but cause additional negative impact in the long term [2]. The long-term negative impact is referred to as the *interest* paid on the debt [1, 9]. So long as the debt is not repaid, the organization pays interest, which likely grows in the future.

Technical Debt can be repaid with a refactoring of the system, to remove the sub-optimal solution: this would prevent the organization to pay a costly interest. However, if the (future) interest, paid on the debt, is not enough to justify the cost of refactoring, there might be no reason to invest resources in refactoring the system. In other words, Technical Debt needs to be prioritized [5] based on a cost/benefits analysis. At the moment, the prioritization of TD is lacking good practices and tools: consequently, TD remains in the systems and causes negative impact. [3, 6].

The interest on the debt, its growth and the cost of refactoring are however difficult to assess [8, 10]. Understanding and communicating with other stakeholders if the Technical Debt is worth refactoring requires assessing interest and cost of refactoring systematically [8]. Also, it is worth understanding when the interest is going to be paid (for example in a few months or many years), to decide when to repay the debt [7]. It is useful to understand where in the system the issues are, and what Technical Debt items are related to each other [5, 11]. However, tool support for such analysis is missing at the moment.

AnaConDebt is a management tool that builds on top of previous empirical research and industrial experiences and aims to support tracking and assessing Technical Debt Items through a TD-enhanced backlog.

2 BACKGROUND OF ANACONDEBT

A portfolio approach to tracking TD is a well-known practice, originally developed by Guo et al. [4]: such practice defines and ranks items in a backlog. Other studies have reported the use of TD backlogs (e.g. [6, 10, 11]). However, existing tools do not use TD cost of refactoring and interest as key parameters. Such features are critical to assess Technical Debt and to decide on its refactoring.

In a previous study [7] the authors developed a method, called AnaConDebt (Analysis of Contagious Debt), together with several industrial partners, to analyze the interest of large Architectural Technical Debt items. The method was found helpful by the practitioners to estimate the impact of Architectural Technical Debt. The tool presented here was called with the same name, AnaConDebt, as it builds on top of such work. However, it presents many differences from the original method, which has been considerably evolved and refined over time according to new experiences gathered from practice.

In a first attempt to create a prototype tool, the original method was modified and simplified to allow the efficient assessment of several kinds of TD items. The prototype was then evaluated, and the positive results were published in a study at the Ninth Workshop on Technical Debt [8]. Such tool allowed the practitioners to assess the negative impact of Technical Debt by systematically assess several aspects, including the future growth of interest. Such prototype was then used multiple times during consultancy assignments with large software companies. This step allowed the author to evaluate and refine the tool further, understanding better the requirements and the useful features.

After the second evaluation, the prototype was found useful by the users; however, the tool needed to be re-implemented to be available as a usable tool. For example, requirements such as portability, security, and usability needed to be taken into account. The author then received funds from the Swedish innovation agency Vinnova, in collaboration with Chalmers Innovation Office, to refactor the tool to make it practically usable and ready for commercialization. The tool is being reimplemented at the time of writing this document and will be launched approximately during spring 2018.

TechDebt '18, May 27–28, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5713-5/18/05…$15.00
https://doi.org/10.1145/3194164.3194185

3 ANACONDEBT

AnaConDebt is a management tool that consists of a TD-enhanced backlog. The backlog allows the creation of TD Items and allows performing TD-specific operations on the items, which are currently not available in other existing tools.

AnaConDebt supports the users in the following activities:

- Tracking Technical Debt Items in a dedicated repository. The TD Items can be further characterized with attributes and properties such as name, description, etc.
- Assessing Technical Debt Principal (cost of refactoring) and Interest (current and future extra-costs) systematically, using an approach that was previously evaluated, both scientifically and in practice. The values are inputted in a simple and intuitive way, using advanced widgets. Additionally, the values and how they are used are explained. This is important, as many tools are difficult to be used by the stakeholders.
- Estimating growth of principal and interest with respect to future scenarios: for example, in short, medium and long term. The time related to the scenarios is customizable.
- Assessing and aggregating the amount of total Technical Debt, Principal and Interest for the whole backlog of Technical Debt Items. This is useful for the practitioners to have an overview a of their Technical Debt.
- Reviewing and assessing previous TD estimations. In fact, it is important to show if the expected principal and interest of Technical Debt were correctly estimated during the initial time of estimation.
- Comparing and ranking Technical Debt Items based on the *convenience of refactoring*, calculated by weighting different parameters related to Principal and Interest. Such parameters are customizable according to the company's context. For example, for some companies the same cost might be more or less important with respect to other factors.
- Visualizing and comparing Technical Debt Items in a cost/benefits graph
- Locating the Technical Debt Items in the system. For example, linking a Technical Debt Item with a specific file or component.
- Visualizing the areas of the system with more Technical Debt Items and with more Interest to pay (more convenient to refactor). This can be useful to group tasks related to the same part of the system.
- Relating Technical Debt Items among each other by defining dependencies. For example, it might be important to specify that one TD item should be refactored before another. Eventually, this feature would help grouping and planning the refactoring of TD Items.
- Thanks to the previous point, calculating an aggregated Principal and Interest based on the dependencies among the Technical Debt Items.
- Presenting an easy-to-understand report on the convenience of refactoring Technical Debt Items to stakeholders that are not technical (e.g. project managers or product owners), to allow an assessment with respect to other project or product aspects.
- Integrating the repository and the TD-specific operation to other tools via a REST API on top of a portable web-service. In particular, the architecture of the tool separates the GUI from a web-service. This way, the data related to Technical Debt can be used in other tools and visualizations: for example, the aggregated data related to the TD intems can be added in a dashboard containing other project- and product data that needs to be compared with the current information on Technical Debt.

4 CONCLUSION

The tool AnaConDebt is the outcome of a research process based on the collaboration with several large software companies and the subsequent transfer of knowledge generating a practical solution. AnaConDebt has been evaluated and re-implemented multiple times and it is now available to improve software engineering practices related to the management of Technical Debt.

5 ACKNOWLEDGMENTS

We thank Vinnova for the Verifiering för tillämpning grant, and to the Chalmers Innovation Office for funding the re-implementation of the tool. We thank Jan Bosch for his continuous support. We thank Enrico Mazzei and the "Mycd di Colli Luca" for the technical assistance and for making it possible to develop the tool. We also thank all the industrial partners of the Software Center for their invaluable insights.

REFERENCES

[1] Areti Ampatzoglou, Apostolos Ampatzoglou, Alexander Chatzigeorgiou, and Paris Avgeriou. 2015. The financial aspect of managing technical debt: A systematic literature review. *Information and Software Technology* 64 (Aug. 2015), 52–73.

[2] P. Avgeriou, P Kruchten, I. Ozkaya, and C. Seaman. Managing Technical Debt in Software Engineering (Dagstuhl Seminar 16162).

[3] Z. Codabux and B. Williams. 2013. Managing technical debt: An industrial case study. In *2013 4th International Workshop on Managing Technical Debt (MTD)*. 8–15. https://doi.org/10.1109/MTD.2013.6608672

[4] Yuepu Guo and Carolyn Seaman. 2011. A Portfolio Approach to Technical Debt Management. In *Proceedings of the 2Nd Workshop on Managing Technical Debt*. ACM, New York, NY, USA, 31–34.

[5] Zengyang Li, Paris Avgeriou, and Peng Liang. 2015. A systematic mapping study on technical debt and its management. *Journal of Systems and Software* 101 (March 2015), 193–220. https://doi.org/10.1016/j.jss.2014.12.027

[6] A. Martini, T. Besker, and J. Bosch. 2016. The Introduction of Technical Debt Tracking in Large Companies. In *accepted at APSEC 2016*. Hamilton, New Zealand.

[7] Antonio Martini and Jan Bosch. 2016. An Empirically Developed Method to Aid Decisions on Architectural Technical Debt Refactoring: AnaConDebt. In *Proceedings of the 38th International Conference on Software Engineering Companion (ICSE '16)*. ACM, New York, NY, USA, 31–40. https://doi.org/10.1145/2889160.2889224

[8] Antonio Martini and Jan Bosch. 2017. The Magnificent Seven: Towards a Systematic Estimation of Technical Debt Interest. In *Proceedings of the XP2017 Scientific Workshops (XP '17)*. ACM, New York, NY, USA, 7:1–7:5. https://doi.org/10.1145/3120459.3120467

[9] Ariadi Nugroho, Joost Visser, and Tobias Kuipers. 2011. An empirical model of technical debt and interest. In *Proceedings of the 2nd Workshop on Managing Technical Debt (MTD '11)*. ACM, New York, NY, USA, 1–8. https://doi.org/10.1145/1985362.1985364

[10] Klaus Schmid. 2013. A formal approach to technical debt decision making. In *Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures*. ACM, 153–162. http://dl.acm.org/citation.cfm?id=2465492

[11] Jesse Yli-Huumo, Andrey Maglyas, and Kari Smolander. 2016. How do software development teams manage technical debt? An empirical study. *Journal of Systems and Software* 120 (Oct. 2016), 195–218. https://doi.org/10.1016/j.jss.2016.05.018