Developing Processes to Increase Technical Debt Visibility and Manageability – An Action Research Study in Industry

Jesse Yli-Huumo[1](✉), Andrey Maglyas[1], Kari Smolander[2], Johan Haller[3], and Hannu Törnroos[4]

[1] Lappeenranta University of Technology, Lappeenranta, Finland
jesse.yli-huumo@aalto.fi, maglyas@gmail.com
[2] Aalto University, Espoo, Finland
kari.smolander@aalto.fi
[3] Tieto Sweden AB, Stockholm, Sweden
johan.haller@tieto.com
[4] Tieto Oyj, Helsinki, Finland
hannu.tornroos@tieto.fi

Abstract. The knowledge about technical debt and its management has increased in recent years. The interest of academia and industry has generated many viewpoints on technical debt. Technical debt management consists of technical and organizational aspects, which make it a challenge in software development. To increase technical debt visibility and manageability, new processes must be developed and thoroughly empirically tested for their applicability. In this paper, we use the action research methodology to design processes for identification, documentation, and prioritization of technical debt. Our partner in this research is a large Nordic IT company Tieto, currently in a need for new ways to improve their technical debt management. The results include a set of processes and templates that were successfully used to identify and document technical debt. The identified technical debt items were later prioritized based on evaluation by Tieto employees. Tieto was able to create a prioritized technical debt backlog, which is now used for reduction activities to create a healthy and sustainable product for the future.

Keywords: Technical debt · Technical debt management · Software process improvement · Action research

1 Introduction

Technical debt refers to a situation in software development where shortcuts and/or workarounds are used in technical decisions to gain time-to-market [1]. The benefit of taking technical debt is an earlier and faster release, which can lead to customer satisfaction and other economic advantages [2]. However, the drawback is the '*debt*' that is left in the system. In the long-term, shortcuts and workarounds will turn to unnecessary complexity (*interest*) in the source code and architecture. Complexities in software can become hard to fix and change, which may cause decrease in software quality and productivity of the development team [3]. Therefore, technical debt can be a major problem for a software development company.

While shortcuts and workarounds can be seen as intentional decisions to speed up release cycles, or to circumvent a complex part of the code, unintentional technical debt occurs without immediate awareness [4]. Unintentional technical debt is introduced to software, for example, by inexperienced developers or legacy software. An inexperienced developer can create technical debt unintentionally with non-optimal solution. Old legacy software can consist of obsolete or non-optimal technology and solutions from past decades, which may require a rewrite or replacement.

Technical debt management refers to activities that are used to manage and reduce both intentional and unintentional technical debt with various approaches, practices and tools [5]. Technical debt management not only includes technical development activities but also organizational ones, such as communication and decision-making.

This study is made in cooperation with one of the largest IT companies in Scandinavia, Tieto. Tieto's Capital Market product unit is currently planning new processes for their technical debt management. The goal of the study is to develop new processes for technical debt identification, documentation, and prioritization. The outcome of this study includes new processes to increase the visibility and manageability of technical debt, which can be used in the future for better decision-making.

This paper is limited to studying technical debt that has already been acquired previously, and does not take in consideration the management activities related to decision-making process of acquiring new technical debt.

2 Background

Processes for technical debt management have been studied and suggested in the literature. Li et al. [5] gathered in a mapping study relevant research on technical debt management. The study showed that technical debt management can be divided into following activities: (1) *identification,* (2) *measurement,* (3) *prioritization,* (4) *prevention,* (5) *monitoring,* (6) *repayment,* (7) *representation/documentation,* and (8) *communication* [5]. Li et al. [5] also state that currently there is a lack of empirical evidence about technical debt management. In this study, we are mainly focusing on three out of the eight management activities. Our goal is to use processes for *representation/documentation, identification,* and *prioritization* of already incurred technical debt to provide empirical evidence with a real case company.

Technical debt representation/documentation has been studied and suggested in literature with specific lists and templates as an approach to store technical debt issues [6, 7]. A backlog or a list should increase technical debt visibility and manageability. When technical debt is properly documented, it is easier to start other technical debt management activities, because it is visible to the company.

Before a technical debt issue can be documented, it has to be identified. Identification of smaller technical debt issues from the source code is possible with specific tools [8]. However, technical debt is not always only related to issues in the source code [9]. Technical debt in software architecture and design is a larger challenge [5, 9].

The identification of architectural technical debt with tools is difficult and often the only solution is to use human knowledge and examination [9].

The prioritization of technical debt is difficult, because some technical debt might be important to fix for business reasons, while other for technical reasons. Some models and methods have been developed for prioritization. Seaman et al. [10] suggested four approaches for technical debt decision-making: *simple cost-benefit analysis, analytic hierarchical process, portfolio management model, and options*. These approaches have been used also in other domains, such as finance [10]. They support evaluating the tradeoffs between proposed enhancements, corrective maintenance, and the payment of technical debt items [10]. Schmid [11] developed a formal model based on providing several well-defined approximations, which can be used for technical debt prioritization. In addition, some papers have used quality attributes from ISO 9126 as an evaluation to technical debt [12–14].

Overall, there exists a variety of different ideas for technical debt documentation, identification and prioritization. However, most of them are focused on one specific activity only. Studies that approach the whole process from identification to repaying technical debt are rare. Therefore, we collaborate with a real software company to find and develop processes, including technical debt identification, documentation, and prioritization. We take inspiration from a study conducted by Li et al. [7] that had a similar goal. Their approach was to identify architectural technical debt based on architecture decisions and change scenarios [7]. Our approach extends this by expanding the technical debt evaluation and prioritization processes. Our goal is to create more reasoning possibilities in decision-making, which is required especially in organizational aspects of technical debt management.

3 Research Methodology

Action research was selected as a research methodology for this study. Action research combines theory and practice [15]. Action research is an iterative process involving researchers and practitioners acting together on a particular cycle of activities, including problem diagnosis, action intervention, and reflective learning [15]. Action research is especially relevant in situations where participation and organizational change processes are necessary [16]. It attempts to provide practical value to the client organization while simultaneously contributing to the acquisition of new theoretical knowledge [17]. The action research cycle [18] consists of three stages: (1) a pre-step - *to understand context and purpose*; (2) six main steps - *to gather, feedback and analyze data, and to plan, implement and evaluate action*; (3) a meta-step - *to monitor*.

The rationale for using action research as a research methodology is the nature of this study. The company in this study had a goal to improve their technical debt management. The research group in this study had previous experience on working with various companies and cases related to technical debt and its management. Therefore, action research, as an approach where both the company and the research group work together to understand the problem and develop a solution, was especially fit for the purpose.

The selected product line in this research is a financial system used in the capital market industry by multiple customers around Nordics. The product is one of the three main products provided by Tieto and it has a long development history including source code from over 20 years ago. The product and development team have faced both technological changes and organizational changes during their lifetime. Now the main objective of Tieto's Capital Market product unit is to migrate to new technology with the aim to replace and rewrite old one, to improve quality and productivity, while still serving all of its customers.

The objective of the study was to increase technical debt visibility and manageability by improving processes related to identification, documentation and prioritization. Therefore, we set up the following research questions to address the problem:

RQ1: How to improve technical debt identification and documentation? The limitations of the tools currently available for technical debt identification can be seen as a big challenge. The identification of architectural technical debt with tools is very difficult. Therefore, most if not all technical debt identifications have to be done with manual code and architecture inspection, where developer or architect examines the system and the source code for possible issues. Our goal is to observe how technical debt is currently identified in practice and how it is documented afterwards. The objective is to identify possible improvements to these current processes, and test them in practice.

RQ2: What factors should be taken in consideration when prioritizing technical debt? The decisions related to technical debt can be sometimes made based on hunches without any specific model or method to follow. Business owners might prioritize issues that give direct value to customers, while technical people might put value more on software quality and sustainability. Understanding both business and technical effects of technical debt repayment can help technical debt evaluation and improve the prioritization process for safer decisions. We will observe the processes of technical debt evaluation and prioritization in practice with the aim to improve technical debt evaluation and prioritization.

4 Action Research Process

The action research process used in this study is presented in Fig. 1. This research can be divided into five main activities and outcomes.

The first step of the research process is *interviews,* where researcher interviews people related to the product line or company to understand the current issues related to technical debt and its management. We conducted seven semi-structured interviews with the average of 45 min. We recorded, transcribed and analyzed all the interviews. In the analysis of the interviews, we identified major issues. First, we did not find any systematic process for technical debt identification, evaluation or prioritization. This led to a technical debt communication gap between the development team and project managers. Knowledge of technical debt seemed to be tacit personal knowledge rather than explicitly stored in a common list. Secondly, we noticed that the developers and architects had much knowledge about the current issues regarding technical debt, but

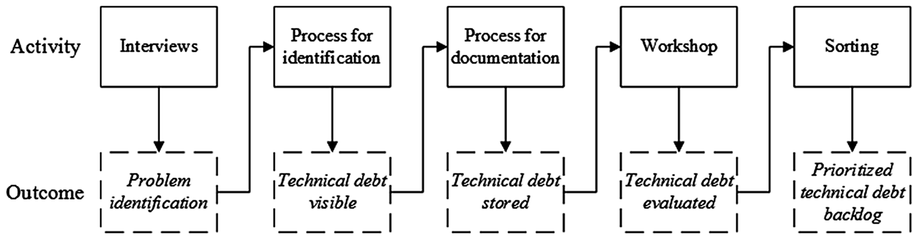

Fig. 1. The research process

there was not any systematic way to document it. Thirdly, when there were technical debt issues in discussion, the decision-making was mostly done based on hunches, rather than evaluating and prioritizing them first. The outcome of this step is a *problem identification,* which helps to understand the problem in current processes within a company.

The second step is to develop a *process for identification.* In our case, the identification was conducted by gathering the data from previous knowledge and history of people related to the product. The members of the product line used ten weeks to search and identify technical debt issues. The reason for manual inspection was that the company did not have any specific tools in use to identify technical debt. The outcome of this step is the increase of *technical debt visibility*, which helps to understand the overall technical debt view.

The third step is to develop a *process for documentation.* We decided to introduce a simple process to document all technical debt issues to a single technical debt backlog. The idea was to use backlog as an aid to make technical debt more visible to everyone in the product line. We used a similar template (Table 1) to Guo and Seaman [6] to collect all technical debt items. The template was sent to nine members of the product line that was later returned back to the managers. The managers then combined all the reported issues and created the technical debt backlog. The outcome of this step is getting *technical debt stored.*

The result of documentation process was technical debt backlog that consisted 47 identified technical debt issues. For categorization we used 15 different technical debt types identified by Alves et al. [19] in a mapping study. The majority of identified technical debt (33/47 issues) was related to issues in *design, architecture, code*, and a new category called *legacy* debt. Other types of technical debt (14/47) *requirements,*

Table 1. Template for technical debt documentation

Technical Debt ID	Technical debt identification number
Date/Reporter	Reporting date/Reporter name
Technical Debt Name	Name of identified technical debt
Description	Description of identified technical debt
Alternatives	Explanation of possible alternative solutions
Rationale	Reasons to fix technical debt

test, test automation, and process debt were associated more to activities outside product implementation.

The fourth step is to organize a *workshop.* We developed a process to prioritize technical debt issues with a simple technical debt evaluation and prioritization template (Table 2). This template was used when all the technical debt issues were collected to the backlog. In the workshop, the participants would evaluate each identified technical debt issue based on the five questions to create a prioritization. The outcome of this step is getting *technical debt evaluated.*

The research group also analyzed the returned evaluation templates based on each question to understand how technical debt was being evaluated.

We identified three different **benefit** categories: *technical, economic,* and *organizational benefits.* Technical benefits include improvements in software *quality,* software *maintainability,* software *reusability,* software *performance,* software *testability,* and software *deployment.* Organizational benefits include better software *deployment,* development team *productivity,* organizational *communicability,* and future *adaptability.* Economic benefits include *economic value* and *customer satisfaction.*

We identified three different **risk** categories: *economic, technical,* and *organizational risk.* Economic risks include *cost, time effort, testing effort,* and *customer satisfaction.* Organizational risks include *management* and *competence.* In addition, technical risks like *system breakdown* and *instability* are critical to companies.

For **reasons**, we identified three different categories: *intentional decision, unintentional cause,* and *organizational cause.* Intentional reasons were often related to *time constraints, lack of resources,* and *business driven development.* Unintentional causes were *legacy product* and *lack of knowledge*. For organizational causes, *software processes* and *lack of management* were the main reasons for technical debt.

We identified two types of **solutions** for technical debt: *technical* and *organizational solutions.* Technical solutions were *refactoring, redesigning, rewriting, architectural analysis,* and *increased testing.* Organizational solutions were *new processes* and *new management plan/strategy*.

The fifth step of the research process is *sorting.* When there is an evaluation for each technical debt item, it is easier to sort the issues out based on their importance. The last outcome of the process is *prioritized technical debt backlog.* The majority of the issues (27/47) were prioritized at the lowest priorities 5 or 4, which shows that most technical debt was not considered dangerous now. There were total of 14 issues rated as the highest priorities at level 1 or 2. There were three level 1 issues related to *legacy*

Table 2. Technical debt evaluation template

#	Question
1	What are the benefits of fixing this issue? (Business value, quality, productivity, less bugs etc.)
2	Are there any risks in fixing this issue? (Expensive, breaks the system etc.)
3	Why was this issue done previously like that?
4	How to fix this issue and what resources the fix would require?
5	From scale 1 – 5, how important would you rank this issue to be dealt with? (1 – most important, 5 – not so important)

debt, which can be explained by Tieto's current goals to migrate to a new technology to replace and rewrite old technology. Interestingly, the priorities also show that most of the technical debt related to *design, architecture,* and *code debt* were prioritized as 4 or 5, while *test, test automation,* and *process debt* were rated higher. However, it is important to notice that number of technical debt issues in *design, architecture,* and *code* is much higher than other types of debts, which might explain the difference.

The outcome of this action research cycle was a prioritized technical debt backlog that can be now used to add more development tasks related to technical debt reduction. For example, Tieto managers expressed that the backlog would be used in the future by Tieto to reduce technical debt in small iterations. Tieto managers also mentioned that this same process would be applied in future to other product lines.

5 Discussion

RQ1: How to improve technical debt identification and documentation? Our study made technical debt identification and documentation possible with simple practices that make technical debt more visible and manageable. These similar practices have been already suggested in other literature [6, 7]. However, the problem is not the practices themselves, but the fact that changing or adding new practices in companies is always a challenge and takes time [20]. In our case, we had a company that was motivated to improve and change these practices. We started some new practices with templates and processes that gathered previously identified technical debt from the minds of architects and developers to a specific backlog designed only for technical debt.

Technical debt identification is a challenge in software development. Identification of smaller technical debt issues happening in single code lines can be done with static code analysis tools and often it can be fixed by single developers. However, larger issues in architecture and structure are often unreachable with tools [9, 21] and require technical knowledge and competence [22], and discussion on an organizational level. In our case, the people in the product line did not use any tools to find and identify technical debt. Instead, the technical debt was identified based on previous experience and history with the product. The experience with the product of workshop participants shows that the people responsible for identification had extensive knowledge of the product development history and high competence to build software. We used this fact to our advantage, since we did not have to guide developers and architects to investigate product history, because the knowledge was already acquired during the development years. This helped to identify existing technical debt and document it based on our recommendations.

An interesting perspective on identified and documented technical debt and is the variety of types of technical debts. The large variety of technical debt types shows that when talking about technical debt, it is not only related to issues in design or code. Instead, like in our case, the same phenomena of shortcuts and bad solutions happen in other parts of software product development as well. To some development teams, technical debt might include only issues happening in the source code and design, while to some other teams, like in Tieto it might also include issues like those in testing and processes. We argue that technical debt management is successful when a company sets a clear standard to what is technical debt in their context and start to manage technical debt based on that standard. However, in academia, there is a need to create a common understanding for technical debt.

RQ2: What factors should be taken in consideration when prioritizing technical debt? The prioritization in this study was based on evaluation of *benefits, risks, reasons, and solutions* of technical debt. Using these factors to evaluate each technical debt issue could be a good beginning for companies that are trying to improve their technical debt management. However, these factors are not always measureable with a numeric value. The *interest* in technical debt that accumulates larger if not repaid, it is difficult to estimate [23]. Therefore, rather than trying to measure exact values, technical debt could be easier to understand from management perspective, if evaluated based on factors related to it.

Companies should evaluate each technical debt issue on the basis of how fixing the issue can benefit both company and software, such as improved quality, and how does this quality improvement affect other factors such as maintainability, performance or customer satisfaction. One challenge and risk of technical debt is that it often requires competence to fix or change existing solutions. When developers are changing very old parts of the code, it is not always certain that it will go as planned and it can be a huge risk that needs to be evaluated before.

Understanding the reasons behind technical debt can help to understand bigger underlying problems with technical debt. For example, a single technical debt issue in one smaller feature can be caused by some larger architectural issue. Instead of just fixing one single technical debt issue with the most economical value, it might be possible that another major technical debt item can be actually more beneficial to fix in a long term. Sometimes the solution might only require small refactoring, while sometimes it might need a full rewrite of that certain part of the code. Therefore, it is important to evaluate how much resources and effort does fixing technical debt require. Understanding the solution can enable a better evaluation, whether the time required for fixing is worth compared to its benefit.

We believe that technical debt prioritization should be done based on evaluation rather than measurement. The combination of the presented factors can be used as a simple way to create basic prioritizations, which can help companies to make decisions with more rationality. The decision-making may improve when development teams and management communicate and understand the benefits and risks in each technical debt issue, accompanied with knowledge on the reasons and solutions for technical debt.

Study limitations. *The first limitation* to this study is the generalization of the results. It is not certain that this process is usable in other companies. In our case, most of the involved people had many years of experience with the product. This helped the identification stage, since the people from the product line had already extensive knowledge about the issues in the product. *The second limitation* is that we conducted only one round of this action research. Conducting more rounds might change some results in the priorities and numbers of technical debt issues, but we believe it would not have any changes to the actual processes that were used in this study. *The third limitation* is that the used process only takes in consideration already occurred technical debt, and does not include management processes for a situation, where a decision has to be made for a new technical debt case. This makes the developed management process limited to only already occurred technical debt.

6 Conclusions

We used the action research process [18] together with a large IT company Tieto to find and develop processes for technical debt identification, documentation, and prioritization to increase technical debt visibility and manageability. The action research process consisted several interviews and meetings with the company representatives and an organized technical debt workshop to improve processes in the company. The outcome of the research was a set of templates and processes to identify, document, and prioritize technical debt. These templates and processes were used successfully at Tieto to transition from a situation where knowledge of technical debt was not explicitly documented, to a situation where a specifically prioritized technical debt backlog was available to reduce technical debt. Tieto's Capital Market product unit is now using this new technical debt backlog to increase technical debt visibility and manageability. Since the results with the developed process were considered successful by both the research group and the company, the same process will be expanded to other product lines in Tieto. The main challenges and lessons learned can be summarized as following:

- Technical debt can be brought visible with simple practices and processes in a company that does not have a priori knowledge on technical debt management.
- Identification of larger scale technical debt, such as architecture and design, with tools is a challenge that needs to be addressed and improved in future research.
- Technical debt documentation can be done with simple templates, but requires motivation and resources from software organization.
- Technical debt prioritization based on measurements is difficult, and therefore rougher evaluations based on e.g. benefits and risks through opinions can be seen easier to start with.

References

1. Cunningham, W.: The WyCash Portfolio Management System, Experience Report (1992)
2. Yli-Huumo, J., Maglyas, A., Smolander, K.: The sources and approaches to management of technical debt: a case study of two product lines in a middle-size finnish software company. In: Jedlitschka, A., Kuvaja, P., Kuhrmann, M., Männistö, T., Münch, J., Raatikainen, M. (eds.) PROFES 2014. LNCS, vol. 8892, pp. 93–107. Springer, Heidelberg (2014). doi:10.1007/978-3-319-13835-0_7

3. Yli-Huumo, J., Maglyas, A., Smolander, K.: The benefits and consequences of workarounds in software development projects. In: Fernandes, J.M., Machado, R.J., Wnuk, K. (eds.) ICSOB 2015. LNBIP, vol. 210, pp. 1–16. Springer, Heidelberg (2015). doi:10.1007/978-3-319-19593-3_1
4. McConnell, S.: Technical Debt-10x Software Development | Construx, 1 November 2007. http://www.construx.com/10x_Software_Development/Technical_Debt/. Accessed 25 March 2014
5. Li, Z., Avgeriou, P., Liang, P.: A systematic mapping study on technical debt and its management. J. Syst. Softw. **101**, 193–220 (2015)
6. Guo, Y., Seaman, C.: A portfolio approach to technical debt management. In: Proceedings of the 2nd Workshop on Managing Technical Debt, New York, NY, USA, pp. 31–34 (2011)
7. Li, Z., Liang, P., Avgeriou, P.: Architectural technical debt identification based on architecture decisions and change scenarios. In: Proceedings of the 12th Working IEEE/IFIP Conference on Software Architecture, WICSA (2015)
8. Zazworka, N., Vetro', A., Izurieta, C., Wong, S., Cai, Y., Seaman, C., Shull, F.: Comparing four approaches for technical debt identification. Softw. Qual. J. **22**(3), 403–426 (2013)
9. Kruchten, P., Nord, R.L., Ozkaya, I.: Technical debt: from metaphor to theory and practice. IEEE Softw. **29**(6), 18–21 (2012)
10. Seaman, C., Guo, Y., Zazworka, N., Shull, F., Izurieta, C., Cai, Y., Vetro, A.: Using technical debt data in decision making: potential decision approaches. In: 2012 Third International Workshop on Managing Technical Debt (MTD), pp. 45–48 (2012)
11. Schmid, K.: A formal approach to technical debt decision making. In: Proceedings of the 9th International ACM Sigsoft Conference on Quality of Software Architectures, New York, NY, USA, pp. 153–162 (2013)
12. Curtis, B., Sappidi, J., Szynkarski, A.: Estimating the size, cost, and types of technical debt. In: Proceedings of the Third International Workshop on Managing Technical Debt, Piscataway, NJ, USA, pp. 49–53 (2012)
13. Theodoropoulos, T., Hofberg, M., Kern, D.: Technical debt from the stakeholder perspective. In: Proceedings of the 2nd Workshop on Managing Technical Debt, New York, NY, USA, pp. 43–46 (2011)
14. Letouzey, J.-L.: The SQALE method for evaluating technical debt. In: Proceedings of the Third International Workshop on Managing Technical Debt, Piscataway, NJ, USA, pp. 31–36 (2012)
15. Avison, D.E., Lau, F., Myers, M.D., Nielsen, P.A.: Action research. Commun. ACM **42**(1), 94–97 (1999)
16. Baskerville, R.L., Wood-Harper, A.T.: A critical perspective on action research as a method for information systems research. J. Inf. Technol. **11**(3), 235–246 (1996)
17. Sjoberg, D.I.K., Dyba, T., Jorgensen, M.: The future of empirical methods in software engineering research. In: 2007 Future of Software Engineering, Washington, DC, USA, pp. 358–378 (2007)
18. Coughlan, P., Coghlan, D.: Action research for operations management. Int. J. Oper. Prod. Manag. **22**(2), 220–240 (2002)
19. Alves, N.S.R., Mendes, T.S., de Mendonça, M.G., Spínola, R.O., Shull, F., Seaman, C.: Identification and management of technical debt: a systematic mapping study. Inf. Softw. Technol. **70**, 100–121 (2016)
20. Dyba, T.: An empirical investigation of the key factors for success in software process improvement. IEEE Trans. Softw. Eng. **31**(5), 410–424 (2005)

21. Zazworka, N., Spínola, R.O., Vetro', A., Shull, F., Seaman, C.: A case study on effectively identifying technical debt. In: Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering, New York, NY, USA, pp. 42–47 (2013)
22. Robillard, P.N.: The role of knowledge in software development. Commun. ACM **42**(1), 87–92 (1999)
23. Falessi, D., Shaw, M.A., Shull, F., Mullen, K., Keymind, M.S.: Practical considerations, challenges, and requirements of tool-support for managing technical debt. In: 2013 4th International Workshop on Managing Technical Debt (MTD), pp. 16–19 (2013)