Introducing a Framework for Managing Technical Debt Developed by Practitioners

Marion Wiese
Universität Hamburg
wiese@informatik.uni-hamburg.de

Abstract

Technical Debt (TD) and the subsequent problems are equally discussed in industry and scientific papers. In this paper we introduce a framework developed by an IT-unit in industry that tried to tackle these problems. The solution consists of methods to manage and repay technical debt. The goal of the framework is to provide a better overview of the TD items for the mangers of the IT-unit. Additionally, the framework is able to differentiate between general technical issues and TD as initially described in [1].

1 Introduction

Technical Debts (TD) describe problems that arise during the further development when workarounds are made due to tight project deadlines or technical improvements are neglected over a longer time. In a technical metaphor to financial debt the workaround is interpreted as the debt and the resulting problems as interest rates. In practice the term is often used for all technical issues even if they don't fit the metaphor.

This paper discusses the approach of an IT-unit where the TD accumulation led to increasing cycle times, more defective systems and discontent of the developers. The enterprise resides in the publishing and advertisement market where the system landscape must be highly flexible to adapt to new marketing ideas. This led to constantly changing systems and tight deadlines to keep up with the market.

The IT-unit is staffed with about 35 people and is composed of three sub units: Business Analysis, Development and Operations.

The Business Analysts (BAs) and even most of the developers were always looking for the quickest solution in anticipatory obedience to an expected timeline. Most of the time the managers of this unit were not included in or informed of the decisions to take on TD. Furthermore, the need to refactor the systems and to repay TD was discounted by the management and the BAs which led to systems that accumulated a large amount of TD.

The IT-unit was confronted with the following (*research*) questions (*RQ*):

RQ 1: How can the TD be repaid enough to reduce the mentioned problems of increasing cycle times, defective systems and discontent of the developers?

RQ 2: How can the more time-consuming technical tasks be included in the TD repayment?

RQ 3: How can an overall view of TD and general technical issues be generated to better manage them and to keep the management informed?

2 Methodology

The framework was developed in a small group of two managers and two solution architects of the IT-unit to tackle the given problems. The development spanned five steps: (I) identification of the causes, (II) identification of management and prevention solutions, (III) assignment of solutions to causes, (IV) development and documentation of the overall framework, (V) establishment of the new processes. The establishment spanned four more steps: (a) introduction to the unit and (b) to the sub units by the managers and (c) introduction to the Scrum teams and (d) support and guidance by the architects. The framework was developed and established two years ago and was under constant refinement by the IT-unit.

3 TD Management Framework

3.1 Identified TD Causes and other Technical Issues

Ten issues were identified that led to the TD in this unit of which not all strictly stick to the metaphor: (1) regularly refactoring of code that may be in a bad condition due to causes that are not specified in detail, (2) workarounds which are made under the pressure of a deadline, (3) implementation that stay incomplete because of changing business decisions, (4) missing documentation or tests mostly due to No. (2), (5) changing of the architecture to fit modern approaches, (6) implementation and refinement of technical monitoring tools, (7) performance optimization, (8) adapting new infrastructure like e.g. version upgrades, (9) proof of concepts (POC) for new tools, (10) bug fixing.

3.2 Identified Solutions

Four solution items for handling these issues were identified and assigned to the respective issues

(*Table* 1). In this framework all other items are called functional requirement tickets (FRT) as they comprise functional requirements. It was decided that the person responsible for an item should be the one that has the most interest in the completion of the item (*Table* 2). Different groups are in charge of filing the tickets (*Table* 2) in the ticket system (*RQ*3).

Cause/Issue	Scope	Solution
(1) Refactoring	small	MT
	big	TP
(2) Workaround		TDT
(3) Incomplete implem.		TDT
(4) Documentation/Test		TDT
(5) Architecture Change	small	MT
	big	TP
(6) Monitoring		MT
(7) Performance	not visible	MT
	visible	FRT
(8) Infrastructure		TP
(9) POC	small	MT
	big	TP
(10) Bugs	not visible/ workaround	MT
	visible	FRT

Table 1: Issues and their assigned solution item

Maintenance Tickets Maintenance tickets (MT) are technically driven tickets that don't have an impact on the user. The goal of the MT is to repay TD continually. Every MT has to contain a one-sentence information that describes the impact of the ticket. This has to be described in a way that is understandable for BAs and managers (*RQ*3). 10% of the planned capacity of every sprint can be invested for MTs according to the architects decision (*RQ*1).

Errors and performance problems can become directly visible to the user. The responsibility for these tickets is assigned according to the visibility which follows the proposed TD landscape discussed in [2].

Technical Debt Tickets A technical debt ticket (TDT) describes a task that is necessary for clean coding and design or to follow the specified architecture while implementing a functional requirement. This task is not necessary for reaching the business goal of the project and can therefore be implemented after a reached deadline. This follows the original definition of TD as proposed by [1]. The goal of the TDTs is to get an overview of the accumulating technical debt of a given project while it is still running and thereby to give the managers an opportunity to intervene.

Technical Projects All tasks that require more than five days development time are handled as a technical project (TP). The goal of TP is to allocate enough time to conduct bigger technical task (*RQ*2). A TP is handled like a business project. Therefore, a technical roadmap has to be constructed and prioritized by the architects which will then be part of the overall project roadmap defined by the managers (*RQ*3).

Solution	Responsible	Filed By	RQ
MT	architect	developers	1,3
TDT	BA	all	1,3
TP	architect	architect	2,3
FRT	BA	BAs	

Table 2: Solution items and their respective responsibilities

4 Conclusion, Study Limitations and Future Work

The presented framework was developed, established and used in practice for two years. It leads to a better overview for the managers and provides an interpretation of the differentiation between Non-TD and TD as mentioned in [3].

There is no proof yet that the approach was successful regarding the RQs. The evaluation to prove the construct and internal validity in form of a structured questionnaire and a statistical evaluation of the ticket backlog is still missing but will be part of the future work of this author. To ensure the conclusion validity the survey will also be conducted at a comparable IT-unit that is not using this framework. The evaluation may also lead to further potential for optimization. To ascertain the external validity, the framework needs to be adapted by other IT-units.

Finally, it needs to be said that the author may be biased towards the framework as she was part of the team that developed and established the framework as an architect of the examined IT-unit and may not have the necessary distance a researcher should possess.

5 Acknowledgments

I would like to thank Gruner+Jahr GmbH - Media Sales Services for their contributions to the framework and for their support of this paper.

6 References

[1] Ward Cunningham. "The WyCash portfolio management system". In: *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA*. Vol. Part F1296. 2. 1992, pp. 29–30.

[2] Philippe Kruchten, Robert L. Nord, and Ipek Ozkaya. "Technical debt: From metaphor to theory and practice". In: *IEEE Software* 29.6 (Nov. 2012), pp. 18–21.

[3] Zengyang Li, Paris Avgeriou, and Peng Liang. "A systematic mapping study on technical debt and its management". In: *Journal of Systems and Software* 101 (Mar. 2015), pp. 193–220.