Technical Debt Quantification through Metrics: An Industrial Validation

Angeliki-Agathi Tsintzira
Department of Applied Informatics
University of Macedonia
Thessaloniki, Greece
angeliki.agathi.tsintzira@gmail.com

Areti Ampatzoglou
Department of Applied Informatics
University of Macedonia
Thessaloniki, Greece
ampatzoglou@gmail.com

Oliviu Matei
R & D Department
Holisun SRL
Baia Mare, Romania
oliviu.matei@holisun.com

Apostolos Ampatzoglou
Department of Applied Informatics
University of Macedonia
Thessaloniki, Greece
ampatzoglou@uom.edu.gr

Alexander Chatzigeorgiou
Department of Applied Informatics
University of Macedonia
Thessaloniki, Greece
achat@uom.gr

Robert Heb
R & D Department
Holisun SRL
Baia Mare, Romania
robert.heb@holisun.com

***Abstract*—Technical Debt is a software engineering metaphor that refers to the intentional or unintentional situation in which a software industry, produces a software at a lower quality, to achieve business goals (e.g., shorten time to market). Nevertheless, similarly to financial debt, technical debt does not come without negative consequences. The accumulation of technical debt leads to additional maintenance. The technical debt metaphor is built around three major notions: principal, interest, and interest probability. The quantification of these notions is the first step towards the efficient management of technical debt, in the sense that "you cannot control what you cannot measure". In this paper, we employ an established method for quantifying technical debt, namely FITTED, to measure the technical debt of an industrial software product, and contrast it to the perception of the software engineers. The main contribution of this work is the validation of FITTED in an industrial setting, and particularly in the Embedded Low Power Systems domain. The results of the study suggest that FITTED is able of accurately ranking software components, with respect to their principal, interest, and interest probability.**

Keywords*—*technical debt, industrial, case study, metrics

I. Introduction

Technical Debt (TD) is a software engineering metaphor that resembles the development of "*poor-quality*" software to going into debt [9]. The rationale behind this metaphor lies on the fact that a company that does not develop a system in optimal quality before release is saving effort (i.e., money). This amount of money (termed ***principal***) increase the capital of the company, and can be invested in any relevant activity, e.g., depositing, development of by-products, etc. Nevertheless, this "*internal loaning*" does not come without a cost. The lowered internal quality of the system, and in particular the lowered levels of maintainability, lead to increased cost while performing any maintenance activity (e.g., adding a feature, resolving a bug etc.). Such additional costs, resemble the payment of the ***interest*** of the loan. In contrast to economics, where the production of interest is a certain event (ruled by interest rate and interest intervals—usually monthly), in TD, interest is produced only if the artifact is being maintained. The probability of an artifact to need maintenance in the future is termed ***interest probability***, and corresponds to the possibility of interest to be produced [1]. The aforementioned terminology is visualized, based on the FITTED framework [2][7] in Figure 1. In Figure 1 the x-axis corresponds to a fitness function that is considered as a proxy of software maintainability, whereas the y-axis corresponds to maintenance effort. For every ***actual*** system, there exists an ***optimal*** one (with higher levels of quality). To reach the optimal system by refactoring the actual one, the development team needs an effort that equals TD principal. In the hypothetical case that the optimal version of the system is maintained, the addition of a feature requires $Effort_{m(optimum)}$, whereas in reality (actual system) the same activity requires $Effort_{m(actual)}$, which is always larger. The difference between the two corresponds to interest.

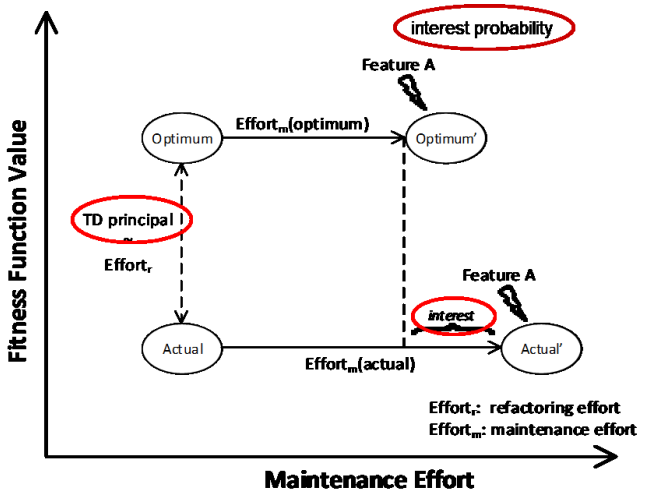

Fig. 1. TD Terminology Visualization

One of the most famous quotes in software engineering, by Tom De Marco [8], suggests that "*you cannot control what you cannot measure*". In that sense, efficient technical debt management [11] cannot be achieved without quantifying the aforementioned notions, upon which the TD metaphor is built. In the literature one can identify a variety of methods for quantifying principal, and substantially less for quantifying interest and interest probability. In this paper, we rely on the technical debt indicators that have been proposed in the context of the SDK4ED project[1], which is a research effort that aims at the provision of a platform for efficient technical debt management in low power software systems, while safeguarding run-time qualities, such as performance, energy efficiency, security, and dependability.

[1] https://sdk4ed.eu/

One of the first activities of SDK4ED project was the development of methods and tools that would be able to quantify the three dimensions of technical debt. The information regarding TD quantification will be presented in the TD dashboard[2], which will be responsible for visualizing the obtained results. As a first step towards the development of the SDK4ED TD dashboard, the consortium has gathered requirements regarding the indicators that shall be presented in the dashboard [5], through a survey with industrial stakeholders. By considering the results of the survey, and the available methods and tools in the technical debt community, TD principal, TD interest and technical debt interest probability were chosen as the indicators to be used in this study (see Section II).

Given the aforementioned means of measuring TD, this paper targets their empirical validation in an industrial setting. Additionally, by considering that the larger the level of granularity the most important the effect of the technical debt item, this paper focuses on the architectural level. In particular, as an architectural unit, we use the software package. Therefore, in this paper we empirically validate the SDK4ED technical debt indicators by performing an industrial case study. To achieve this goal, we use the SDK4ED analysis methods to calculate principal, interest, and interest probability for the packages of a system, and then contrast the obtained ranking with the perception of industrial stakeholders, on the same aspects.

The rest of the paper is organized as follows: in Section II, we present background information, whereas in Section III, we introduce the study design. Next, in Section IV, we present the corresponding results, which are discussed in Section V. The paper is concluded in Section VI.

II. Background Information

In this section we present information on the principal calculation, as executed by SonarQube and used in this study, as well as on the methodology proposed by FITTED framework for the estimation of interest. Moreover, we refer to the notion of interest probability, as it is captured by the concept of change proneness.

- ***TD principal*** is relatively easy to identify, quantify and monitor. In most approaches, principal is calculated by summing up the estimated effort to resolve every single defect that is identified through automated tools. As described in Figure 1 above, in the FITTED framework [7], we have assumed a software system, with an actual design quality, estimated by the use of a fitness function. The effort needed to convert the current system into one with optimum design quality represents the TD principal. In the present study, TD principal has been selected to be quantified through SonarQube as the effort to fix all inefficiencies [10]. SonarQube is based on the SQUALE method and: (a) contrasts the source code of an application with a set of predefined rules, so as to identify violations, and (b) for each identified violation it calculates a remediation time that is required to resolve it. The sum of the remediation time for all identified violations is recorded as the SQUALE index, representing TD principal. SonarQube has been executed in its default configuration. The conversion of the default representation of minutes as provided by SonarQube to USD currency, has been made by using a default hourly rate of 45.81$.
- ***TD interest*** is the most emphatic financial term that is used in TDM research [1], [11] and it is calculated in various ways [2]. In this paper, TD interest is calculated based on the FITTED framework [2], [7]. FITTED is a framework for managing interest in technical debt, borrowing the rationale of equilibrium achievement in economic theory. The framework has been originally introduced to assess the sustainability of a software system, i.e., the period in which the cumulative interest is lower than the saved principal [2]. To achieve this goal, FITTED proposed a methodology for assessing TD interest, primarily based on the definition of interest as the difference in maintenance effort between an optimal and an actual (non-optimal) system or artifact (see Figure 1) [7]. Moreover, since interest is closely related to maintainability [3], FITTED framework proposes the calculation of interest based on well-known object-oriented maintainability predictors [12]. More specifically, FITTED suggests the following steps: (a) the identification of five artifacts that are structurally similar to the artifact under consideration; (b) based on the values of the selected object-oriented metrics for all structurally similar artifacts, compile an artificial optimal one; (c) calculate the average distance of the artifact under analysis from the artificial optimal one—this distance is referred as the ratio of additional maintenance effort; (d) calculate the average maintenance product (i.e., lines of code maintained) in each version; (e) multiply the ratio of additional maintenance effort with the average maintenance product; (f) divide the previous outcome with the average lines of code maintained in one hour, so as to retrieve the interest in minutes; and (g) calculate interest in currency using the same hourly rate as in principal calculation [4].
- ***Interest Probability*** is usually referred to in technical debt literature as the probability of interest to occur, which depends on the probability of a TD artifact to change in the next versions of the system [2]. The quality property that is closer to this concept is change proneness. Since interest accumulates during maintenance activities, change-prone classes are considered more possible to incur interest than less change-prone ones [2]. As an indicator for change proneness we use Module Change Proneness Metri (MCPM) [6], which calculates the probability of a component to change due to internal (i.e., structural) or external (e.g., changes in requirements) reasons. The calculation of the metric considers not only the change history of the component, but also the structural dependencies, which can lead to ripple effects [5].

III. Case Study

In this section we report the case study protocol which is designed based on the guidelines of Runeson et al. [13], and reported given the Linear-Analytic Structure. In particular, in the forthcoming sections we report the research questions, the cases and units of analysis selection, and the data collection and analysis methods.

[2] https://sdk4ed.se.uom.gr/

A. Research Questions

The high-level goal of this case study is to validate the SDK4ED technical debt indicators in an industrial setting. To achieve this goal, we have split this high-level goal to three research questions.

RQ$_1$: What is the accuracy of principal estimation?

RQ$_2$: What is the accuracy of interest estimation?

RQ$_3$: What is the accuracy of interest probability estimation?

B. Cases and Units of Analysis

This study is an embedded multiple case study, in which the case is an existing software system (written in Java), and the units of analysis are its packages. The system that we have analyzed is MaQuali that is developed by Holisun SRL. ***MaQuali*** is a software application developed for serving as a quality management system (ISO 9001), along with handling business processes. It consists of 990 classes (152K lines of code) that have been developed between 2009 and 2018. The system consists of 6 main modules, managing the following entities: (a) fiches of progress, (b) actions to be taken, (c) documents involved in ISO quality control, (d) planning, (e) useful information, and (f) milestones.

C. Data Collection

To answer the aforementioned research questions, we have performed a two-step process. First, we analyzed the MaQuali source-code base with the SDK4ED toolkit and quantified the three aspects of technical debt: TD principal, TD interest, and TD interest probability for every package of the software. Afterwards, we ranked the packages with respect to the principal and then we have demarcated 10 areas, each one containing the 10% of the packages. Next, we selected ten software packages, randomly picked from each one of the 10% areas. This process has led us to the following dataset.

TABLE I. TD ASSESSMENT OF INIDCATIVE MAQUALI PACKAGES

Package	Principal	Interest	Interest Probability
fr.icms.db	12,476.35 $	57.67 $	0.93
fr.icms.sorters	70.24 $	0.00 $	0.10
fr.icms.models	694.02 $	16.22 $	0.86
fr.icms.streams	61.84 $	0.18 $	0.85
fr.icms.mail	203.09 $	16.51 $	0.93
fr.icms.renderers	2,283.63 $	0.46 $	0.86
fr.icms.printing	847.49 $	1.72 $	0.92
fr.icms.graph	2,135.51 $	0.70 $	0.92
fr.icms.ui	1,162.05 $	9.97 $	0.83
fr.icms.os	371.82 $	4.56 $	0.93

As a second step, we asked the software engineers of Holisun that focus on MaQuali maintenance to rank the aforementioned packages in three dimensions, based on the following questions:

- Please rank the aforementioned packages (ties are acceptable—however, not preferable) in terms of their level of quality (e.g., coding standards, maintainability, coding violations, etc.). In other words, rank with 1 the package that you would need more time to refactor so as to improve its quality, and with 10 the package, whose fixing will be trivial, since it does not suffer from many problems.
- Please rank the same packages (ties are acceptable—however, not preferable) in terms of how frequently you need to change them. Assign 1 to the package that changes more frequently and 10 to the package that changes less frequently or not at all.
- Please rank the same packages (ties are acceptable—however, not preferable) in terms of the total effort that you spent for their maintenance. As maintenance, please consider the time that you spend for adding a new requirement, for fixing a bug, etc. In this question, consider not only the time required for one maintenance action, but also how frequently you need to maintain them. Nevertheless, this question is not identical to the previous one, in the sense that a package might change often, but you change just a few lines, whereas there are other that change rarely, but when they do, a major re-writing is required. Please consider the total maintenance effort for this question. Assign 1 to the package that requires the most maintenance effort and 10 to the package that requires the least maintenance effort.

To remind the functionality that each package provided to the system, the software engineers are given some indicative classes for each package. In every question, the packages have been shuffled, and of course the assessments of each package, based on the SDK4ED platform have been hidden. The analysis of the respondents' answers (5 software engineers) have been aggregated, and led to our dataset, in which each row represented a package, whereas the columns held the following information:

V1. Package Name

V2. Indicative Classes

V3. SDK4ED Principal Assessment

V4. SDK4ED Interest Assessment

V5. SDK4ED Interest Probability Assessment

V6. Perceived SDK4ED Principal

V7. Perceived SDK4ED Interest

V8. Perceived SDK4ED Interest Probability

D. Data Analysis

The aforementioned data have been analyzed using descriptive statistics and by Spearman Correlation in pairs. Each pair consists of the SDK4ED assessment and the perception of the stakeholders, for each aspect of technical debt (e.g., V3 against V6).

IV. RESULTS

The dataset that we have obtained, after transforming absolute values to rankings, is presented in Table II. We note that for aggregating the individual responses of stakeholders to the one presented in Table II, we have used the mode function. In case of tie, we used the 2nd most frequent value for sorting.

TABLE II. RANKING BASED ON SDK4ED AND STAKEHOLDERS

Package	SDK4ED			Perceived by Stakeholders		
	Principal	Interest	Interest Prob.	Principal	Interest	Interest Prob.
A	1	1	1	1	1	1
B	5	6	5	8	4	2
C	8	2	2	4	7	3
D	7	5	3	7	6	4
E	6	3	6	6	2	5
F	3	7	4	2	8	6
G	9	10	10	9	9	7
H	10	9	8	10	10	8
I	2	8	7	3	5	9
J	4	4	9	5	3	10

A graphical representation of the results of Table II is provided in the scatterplot of Figure 2. The x-axis in the figure corresponds to the rank of the package, based on the SDK4ED platform assessment, whereas the y-axis the assessment, based on stakeholders' expert opinion. In order for the two opinions to fully match, the points shall fall into main diagonal.

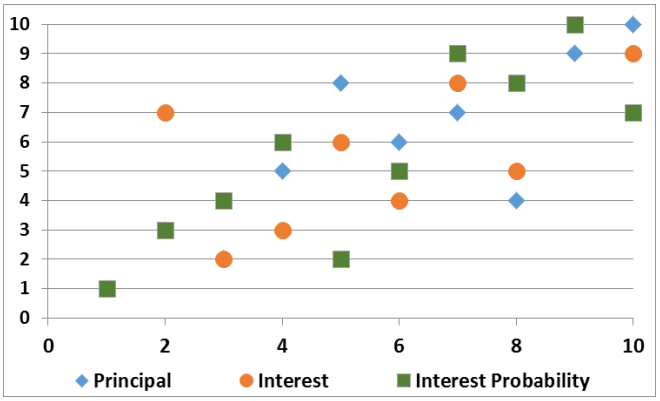

Fig. 2. Scatterplot on the Aggrement between Stakeholders Expert Opinion and SDK4ED platform assessment

Although, visually, the majority of the points approximate the main diagonal line, to draw safer conclusions, we have performed the Spearman correlation analysis. The results of the analysis are presented in Table III. In particular, for every pair we report: (a) the correlation coefficient, and (b) the level of statistical significance. In the table, with bold fonts we designate statistically significant correlations at the 0.05 level, whereas with italic fonts, correlations that are statistically significant at the 0.01 level.

The results of Table III suggest that the SDK4ED platform is able to accurately quantify the basic concepts of the TD metaphor, namely principal, interest, and interest probability. Among those concepts the one for which the correlation between the practitioners' expert opinion and the platform assessment is stronger is principal, followed by interest probability. Additionally, we can observe that some concepts as perceived by practitioners are related: for example, the ranking of packages in terms of interest and interest probability are strongly correlated at a statistically significant level.

TABLE III. CORRELATION ANALYSIS

Practitioners' Opinion	Score	Practitioners' Opinion		
		Principal	Interest	Interest Probability
Principal	Correlation Coefficient		,370	,345
	Sig. (2-tailed)		,293	,328
Interest	Correlation Coefficient	,370		*,697*
	Sig. (2-tailed)	,293		*,025*
Interest Probability	Correlation Coefficient	,345	*,697*	
	Sig. (2-tailed)	,328	*,025*	

Practitioners' Opinion	Score	SDK4ED Assessment		
		Principal	Interest	Interest Probability
Principal	Correlation Coefficient	**,830**	*,673*	,115
	Sig. (2-tailed)	**,003**	*,033*	,751
Interest	Correlation Coefficient	,552	*,733*	,576
	Sig. (2-tailed)	,098	*,016*	,082
Interest Probability	Correlation Coefficient	,576	,321	*,818*
	Sig. (2-tailed)	,082	,365	*,004*

V. DISCUSSION

In this section we discuss the main findings of the paper, by first interpreting them, and next by providing useful implications to researchers and practitioners.

Assessment Capacity of SDK4ED platform. The results of our industrial case study suggest that the provided indicators are strongly to very strongly correlated with the underlying concepts. The relation between the indicators and the concepts is very strong for principal and interest probability, and strong for interest. This finding can be considered as expected in the sense that interest is the vaguest concept in the TD metaphor, since it involves a lot of uncertainty. In particular, interest does not only rely on structural aspects of the software, but also on the extent to which the artifact is being maintained. On the other hand, principal assessment only relies on structure, whereas interest probability relies heavily on historical data.

Interrelations among Concepts. The analysis has pinpointed that some TD concepts are interrelated, both in terms of practitioners' perception and in terms of indicators. First, the indicators for principal and interest seem to be (at least) strongly correlated to the perception of stakeholders' on principal. This suggests that the two concepts are themselves related. This is an expected outcome, since even in traditional economics the terms of principal and interest are analogous. In TD, although the term of interest rate is not defined, there seems to be an underlying relation, confirming the quote that "*the poor is getting poorer*".

Furthermore, interest is strongly correlated to interest probability. This is an expected outcome, in the sense that the more frequent the accumulation of interest is, the larger the amount that is accumulated. This relation has been made explicit to the practitioners while stating the 3rd question of the study, validating that this aspect of interest has been taken into account in their answers.

Implications to industry and the academia: Given the above, we suggest industrial stakeholders to exploit the

SDK4ED TD dashboard as part of their quality assurance processes, since it seems to be accurately reflecting their own perspective. Such a use would be useful especially for novice practitioners who do not have the experience to quickly and accurately judge the aspects of TD. Regarding academic purposes, we encourage the further investigation of TD interest phenomenon, and especially its quantification. Also, studying the relation between interest and principal seems as a promising research opportunity. Finally, the academic community can exploit the conclusions of this study towards the direction of familiarizing software engineering students with the concept of technical debt and enhancing their awareness on the importance of TDM to the system's sustainability.

VI. Conclusions

Considering the importance of quantifying TD principal, TD interest and interest probability for the effective exercise of technical debt management, our study validates the existing FITTED framework in an industrial environment. The methodology of FITTED is being used to quantify TD principal and interest. The results of our study suggest that there is a strong correlation between the indicators used and the concepts, as perceived by the developers/participants. Moreover, our results point out correlations: (a) between the notions of interest and principal, and (b) between those of interest and interest probability. Given the aforementioned discussion of the results, implications to both practitioners and academia have been suggested. On the one hand, considering industry, we propose ways for the efficient management of technical debt, and, on the other hand, regarding academics, we propose tentatively interesting future work opportunities: further investigation of TD interest, and the enhancement of TD awareness for young software engineers.

Acknowledgment

Work reported in this paper has received funding from the European Union Horizon 2020 research and innovation programme under grant agreement No. 780572 (project: SDK4ED).

References

[1] Ampatzoglou, Ar., Ampatzoglou, Ap., Chatzigeorgiou, A. and Avgeriou, P., "The financial aspect of managing technical debt: A systematic literature review", Information and Software Technology, 70, pp. 100-121, 2015

[2] Ampatzoglou, Ar., Ampatzoglou, Ap., Avgeriou, P., and Chatzigeorgiou, A., "Establishing a framework for managing interest in technical debt," 5th International Symposium on Business Modeling and Software Design (BMSD), Milan, Italy, 2015

[3] Ampatzoglou, Ar., Ampatzoglou, Ap., Chatzigeorgiou, A., Avgeriou, P., Abrahamsson, P., Martini, A., Zdun, U., and Systa, K., "The perception of technical debt in the embedded systems domain: an industrial case study" 8th International Workshop on Managing Technical Debt (MTD'16), IEEE, October 2016, Raleigh, NC, USA.

[4] Ampatzoglou, Ar., Michailidis, A., Sarikyriakidis, C., Ampatzoglou, Ap., Chatzigeorgiou, A., and Avgeriou, P., "A framework for managing interest in technical debt: an industrial validation", 2018 International Conference on Technical Debt (TechDebt 2018), ACM, May 2018, Gothenburg, Sweden.

[5] Arvanitou, E. M., Ampatzoglou, A., Chatzigeorgiou, A., and Avgeriou, P., "A Method for Assessing Class Change Proneness", 21st International Conference on Evaluation and Assessment in Software Engineering (EASE' 17), ACM, 15-16 June 2017, Sweden

[6] Arvanitou, E. M., Ampatzoglou, A., Tzouvalidis, K., Chatzigeorgiou, A., Avgeriou, P. and Deligiannis I., "Assessing Change Proneness at the Architecture Level: An Empirical Validation", 1st International Workshop on Emerging Trends in Software Design and Architecture (WETSoDA' 17), Nanjing, China, 4 December 2017.

[7] Chatzigeorgiou, A., Ampatzoglou, Ar., Ampatzoglou, Ap., and Amanatidis, T., "Estimating the Breaking Point for Technical Debt", 7th International Workshop on Managing Technical Debt (MTD '15), IEEE, Computer Society, 2015.

[8] DeMarco, T., "Controlling Software Projects: Management, Measurement, and Estimates", Prentice Hall; 1st Edition, 1986.

[9] Kruchten, P., Nord, R. L., and Ozkaya, I., "Technical Debt: From Metaphor to Theory and Practice," IEEE Software, vol. 29, no. 6, pp. 18–21, Nov. 2012

[10] Letouzey, J.-L., "The SQALE Method for Managing Technical Debt", 3rd International Workshop on Managing Technical Debt (MTD'12), Zurich, Switzerland, 2012, 5 June 2012

[11] Li, Z., Avgeriou, P., and Liang, P., "A systematic mapping study on technical debt and its management", Journal of Systems and Software, ACM, 101 (3), 2015, pp. 193-220

[12] Riaz, M., Mendes, E., and Tempero, E., "A Systematic Review of Software Maintainability Prediction and Metrics", 3rd International Symposium on Empirical Software Engineering and Measurement, Lake Buena Vista, FL, USA, 15-16 Oct. 2009

[13] Runeson, P., Höst, M., Rainer, A., and Regnell, B., "Case Study Research in Software Engineering: Guidelines and Examples", John Wiley and Sons, Inc, 2012.