Usability Technical Debt in Software Projects: A Multi-Case Study

Luiz Carlos da Fonseca Lage
Computing Institute
Fluminense Federal University
Niteroi, Brazil
luizlage@id.uff.br

Marcos Kalinowski
Informatics Department
PUC-Rio
Rio de Janeiro, Brazil
kalinowski@inf.puc-rio.br

Daniela Trevisan
Computing Institute
Fluminense Federal University
Niteroi, Brazil
daniela@ic.uff.br

Rodrigo Spinola
Systems Graduate Program
Salvador University
Salvador, Brazil
rodrigo.spinola@unifacs.br

***Abstract*—Background: Over the years, several studies were conducted aiming at understanding the Technical Debt (TD) phenomenon and its implications on software development. Most of these studies focus on source code related TD types. The absence of empirical studies on usability debt motivated our research. Aims: The goal of this paper is to provide an initial usability debt characterization in software projects regarding its occurrence, type, and resolution effort. Method: We conducted a multi-case study, analyzing TD items of five software projects from four different companies. Results: We identified and classified 145 TD items in the projects. The analysis of these items allowed us to observe that the frequency of usability TD items ranged from 10.4% to 20.8% in the projects. The usability debt items cover a range of usability issues, violating eight out of the ten Nielsen usability heuristics. Regarding effort for paying the TD, usability debt items require a relatively low effort, ranging from 5.1% to 6.7% of the total TD resolution effort in the analyzed projects. Conclusions: Usability TD items are frequent, concern relevant usability issues and typically require low effort for their payment. Hence, paying this type of TD should receive high priority in TD management strategies.**

***Index Terms*—Technical Debt, Usability Debt, Case Study.**

I. Introduction

Software development teams often face the challenge of delivering software products under tight schedules, while attempting to maintain quality. To deal with constraints of time and resources, software developers often need to prioritize by focusing on crucial requirements and even taking shortcuts [6]. In addition, during development, software quality tends to decline when considering the internal software structure, adherence to standards, documentation, and ease of understanding for future maintenance [8]. In this context, Cunnigham [3] presented a comparison of technical complexity and debt and stated that: "delivering an immature code is like getting into debt". This allusion to financial debt coined a new concept in the engineering of software called Technical Debt (TD).

The TD concept refers to the consequences of weak software development [16], usually resulting from the delay or shortcut of development and maintenance tasks, to take advantage of the schedule or, unintentionally, due to developer failures [6]. Thus, it is possible for TD to bring a short-term benefits in terms of higher productivity and less effort, but leaving a debt that may have to be adjusted with interest at a later date [1] [7] [15].

978-1-7281-2968-6/19/$31.00 ©2019 IEEE

Several studies have emerged seeking to understand the concept of TD and its implications for software development, including the generation of evidence and the creation of taxonomies [1] [2] [5] [7] [9] [13] [16]. In a recent and extensive tertiary study, Rios *et al.* [13] identified several types of TD. In our work we will consider this taxonomy and focus on usability debt, defined as "inappropriate usability decisions that will need to be adjusted later".

Several authors recognize importance of usability debt in practice [12] [17]. Zazworka [17], for example, assert that: "Tools can support the identification of defect and design debt in a project, but not other types of debt that have been encountered by developers, for example: documentation and usability debt.". However, we could not find any studies that have this type of debt as the main focus of their investigation. The objective of this paper is to investigate the usability debt phenomenon in order to fill this gap and to shed some light on its typical occurrence, nature, and resolution effort.

Therefore, we conducted a multi-case study in five software projects of four different companies to investigate the occurrence of usability debt in practice. Those projects concerned different application domains, team sizes, development methods and development technologies. Each case study involved authorized access to corporate management systems and several meetings with the project teams, making use of triangulation of data sources.

Our main findings indicate that usability debt is common in software projects. Regarding the type, we classified the usability debt items according to the violation of Nielsen's usability heuristics [11], and observed that our sample of TD items concerned violations of eight different heuristics. Finally, we observed that usability TD items typically require a low resolution effort, when compared to other TD item types. Hence, usability debt showed being frequent, affecting user satisfaction, and requiring low resolution effort. As a consequence, we put forward that practitioners should consider this type of TD in their TD management strategies.

The remainder of this paper is organized as follows. Section 2 describes the multi-case study. Section 3 provides details on how the data collection procedures were applied within

the different projects. Section 4 presents the data analysis and answers the research questions. Section 5 discusses threats to validity. Section 6 contains the concluding remarks.

II. Multi-Case Study

In this section our multi-case study is described, following the guidelines provided by Runeson *et al.* [14].

A. Research Questions

This paper aims to answer the following Research Questions (RQs):

RQ1: Are usability debt items common in software development projects? The answer to this question aims to provide the frequency of usability TD items in each project, when compared to the total amount of TD items.

RQ2: What kinds of usability debt tend to occur in software development projects? To answer this question, after identifying the usability debt items, we classified them based on Nielsen's usability heuristics [10] [11].

RQ3: What is the effort to resolve usability debt items in software development projects? To answer this question, we first interacted with the project teams to gather the total effort (measured in man hours) for paying the TD items off. From this effort, we then analyzed the total effort for TD resolution, in relation to the effort invested in the specific resolution of usability debt.

B. Selection of software projects

Five software projects from four public companies were selected. As contextual information is very important for case study research [14], for each company we will describe the following items: company summary description; Project summary description; Project team; Project development method and technologies; and Project status.

1) Company-1 (TCE-MT): **Company description.** This company is from the public-sector and has more than 60 years of existence. It controls the management of public resources/budgets of the state of Mato Grosso in Brazil.

Project description. From this company, two software projects were selected, which we will be named here as Project-A and Project-B. The purpose of **Project-A** is to automate the process for generating audit reports. The system serves more than 100 auditors and is considered by the company as large and highly complex. The goal of **Project-B**, which is also considered by the company as large and complex, is to support the inspection and detection of companies providing services within the state that did not send documents and information required by law.

Project team. The technical team of project-A has 20 professionals working in the following roles: Project Manager (1), BI Analyst (3), Requirements Analyst (2), Systems Analyst (5), Developer (5), Quality Analyst (4), Support Analyst (6), and Infrastructure Analyst (2). The technical team of Project-B has 22 professionals working in the following roles: Project Manager (1), BI Analyst (3), Systems Analyst (5), Developer (1), Database Analyst (2), Quality Analyst (6), Support Analyst (3), and Infrastructure Analyst (3).

Project development method and technologies. The company has its own development methodology, which is essentially plan-driven and based on the Rational Unified Process. This methodology is used in both projects. The issues of the projects are recorded in a change management system called Redmine. Regarding the development technologies, both systems were developed as web applications in Java and based on the MVC framework, using Oracle 11g database and JBoss EAP application server.

Project status. The development of Project-A began in 2012 with the first delivery in production in 2014. The development of Project-B also began in 2012 and the first delivery in production happened in 2013. Both projects are in production and under development (maintenance and new functionalities).

2) Company-2 (ALMT): **Company description.** This company is also from the public-sector and of the state of Mato Grosso in Brazil. It represents an agency of the legislature, with more than 180 years of existence that acts in the creation of laws for the state and supervision of the state executive power.

Project description. In this company, the project under study will be referenced as **Project-C**. This project is composed of a set of Web systems (Intranet, Web site, Human Resources, among others). The project undergoes continuous evolutionary maintenance, adding new features to the interrelated systems.

Project team. The project team consists of ten professionals working in the following roles: Project Manager (1), Developer (7), Designer (1), and a Tester (1).

Project development method and technologies. This project uses the Scrum agile methodology. The issues for the projects are analyzed, classified and inserted into a backlog until they are effectively prioritized and executed. Currently, the team uses two open-source tools to support change management: GLPI and OpenProject. Regarding the development technologies, the project uses PHP with the Symfony Framework and MySQL databases.

Project status. The development of the project began in early 2008 and it went into production at the end of 2008. It is still in production and development (maintenance and new functionalities).

3) Company-3 (ADDLabs): **Company description.** This company regards a Research and Development (R&D) laboratory of the Fluminense Federal University. It conducts research and develops technologies, since 1996, in Artificial Intelligence and Human-Computer Interaction, mainly for contractors from the Oil&Gas sector.

Project description. The project under study in this company will be referenced as **Project D**. This project concerned the development of a prototype, contracted by a large oil company, to support the evaluation of the consistency of engineering flow charts, instrumentation design models and other specific 3D models.

Project team. The project team was composed of seven professionals, working on the following roles: Project Manager

(1), Business Analyst (1), System Analysts (2), Developer (2), and Tester (1).

Project development method and technologies. The project uses Scrum, with active backlog management of issues using a customized product backlog spreadsheet and Trac as change management system. The development technologies involved Rational Rose for UML modeling, and Visual C++ for programming.

Project status. The development of Project-D began in December 2012 and the software went into production in June 2014. It is still in production and under development (mostly maintenance).

4) Company-4 (UFMT): **Company description.** This company refers to the IT area of Federal University of Mato Grosso. This area is responsible for developing and maintaining academic administrative systems for the university.

Project description. The project under study in this company will be referenced as **Project E**. This project is made up of demands from a set of related academic web systems.

Project team. The project team has twelve employees, working in the following roles: Project Manager (1), System Analysts (5), Developer (5), and Tester (1).

Project development method and technologies. The methodology used for system development was defined by the company based on concepts from the Rational Unified Process. The change management system used is RedMine. The development technologies involve C# technologies and the .NET framework, using SQL Server 2008 database and IIS application server.

Project status. The development began in 2001 and the first versions went into production in 2002. Since then the project is under corrective and evolutionary maintenance.

It is noteworthy that our sample of projects comprises different application domains, team sizes, development methods (e.g., two companies were mainly agile and two were plan-driven), and development technologies (for instance, different frameworks and base programming languages: Java, PHP, C++, C). This diversity was intended for our multi-case study investigation, which focuses on the usability debt phenomenon and should not be biased by specific project settings.

C. Procedures for data collection

All projects followed the same procedure, involving the following five steps:

Training the technical team: The goal of this step was to train the teams regarding TD concepts and the adopted TD type taxonomy [13]. The training was provided by one of the authors, at the workplace of the participating organizations.

Preliminary selection of TD items: The objective of this step was to make the first selection of issues with strong indications of TD (i.e., items that arose due to development shortcuts or to unintentional defects that are known by the team but had their fixing deliberately postponed [16] [6]), according to the authors and the project manager, and classify them by type, according to the adopted taxonomy. At the end of this step, a list must be generated with selected and sorted candidate TD items.

Evaluation and validation of TD items: The objective of this step is for the technical team (at least three representatives for each project) to meet in order to analyze and validate the previously selected TD items. At the end of this step, the TD item list must have been validated.

Closing of the list of TD items: The objective of this step is (for the project manager and the authors) to conduct a meeting to approve the previously validated TD item list, clarifying any remaining doubts and finalizing the TD item list.

Estimation of the effort to pay off the TD: At this step, the project manager uses his expertise to estimate the effort (in "man hours") to pay TD items. The managers used their already established estimation methods. At the end of this step, the selected TD items should have their estimated efforts.

III. Data Collection

In this section we detail how the data collection procedures were applied in the different companies and projects.

A. Training of the technical team

The companies made the involved IT professionals available to participate in our study. The term "Technical Team" refers to these IT professionals. In general, after the training, professionals assimilated the concepts and taxonomy very well. A positive consequence of the training was improving the awareness of the technical teams on the importance of identifying and managing TD.

B. Preliminary selection of TD items

The standard filter used prior the selection of candidate technical debt items in all projects was as follows: items registered in their respective repositories, dated before 2017 and with "open" status at the time of selection.

One of the authors had access, with a read only profile, to the change management systems of projects A, B and D. The initial selection was conducted by that author and validated by the remaining authors and the respective project managers, who also supported the classification. For projects A, B and D the number of selected (and classified) items was, respectively, 120, 24 and 64.

In projects C and E, the preliminary selection was left to the respective project managers, as the first author was not authorized to directly access the data. In total, 27 and 20 items were selected. These items were then classified by the authors with the support of the respective project managers.

In total, 255 TD items were selected and classified. It is important to state that the project manager and the technical team did not focus on any specific type of TD item during this step and that they did not know at any point of the case study that our focus would be on usability debt.

C. Evaluation and validation of TD items

In this step, at least three representatives of each projects technical team met to analyze the selected and classified TD items, making the adjustments that they deemed necessary. I.e., during this process, the technical team had the prerogative to include, change or delete items, and even reclassify them, according to their understanding, always having as reference the adopted taxonomy.

In **Project-A**, this phase was carried out in a very detailed way. The technical team first analyzed the classification of all candidate TD items. Thereafter, the focus was on analyzing the actual status of the items within the context of the project. The technical team understood that there were too many classified items that should not be there. They excluded items that were: (a) finalized as part of other demands, (b) canceled by a higher management decision, or (c) outside the scope of the project (obsolete). Using this criterion, 26 out of the 120 items, which proved to represent actual pending TD items, remained. A possible explanation for the high number of eliminated items may be redundant issues in the change management system, and not updating the status of canceled or obsolete issues. Most importantly, our final sample included only validated and real pending TD items.

In **Project-B**, of the same company, the preliminary selection was conducted more carefully, considering lessons learned from Project A. The list containing 24 TD items was sent to the technical team for validation. After the analysis and discussions, all 24 items were validated.

In **Project-C** the technical team only changed the classification of one item, validating all 27 items as TD items.

In **Project-D**, the meeting of the technical team members resulted in 16 items being excluded and 11 being reclassified, generating a new list with 48 TD items. According to the project manager, the 19 items were excluded because they were already paid off or for being obsolete for the project.

In **Project-E**, after the validation meeting, the technical team kept the 20 TD items of the original list unchanged.

At the end of the validation step, the total of selected items went from 255 to 145 items.

D. Closing the list of TD items

At this stage, one of the authors held final meetings with each project manager and selected members the technical team. All previously selected and evaluated items of the projects were validated as pertinent and correctly classified, further improving our confidence in the TD item selection and evaluation steps.

E. Estimation of the effort to pay off the TD

At this step, the project managers should use their expertise to estimate the effort (in man hours) to pay off the TD items. Project managers used their already established estimation methods.

For Project-A, as done for estimating other development efforts, the manager divided the TD items into six generic activities and valued man hours for these activities, asking for the help of technical team members to improve the precision of his estimates whenever needed. For each TD item he valued the following generic activities: (i) analysis and design, (ii) architecture and database, (iii) development, (iv) tests, (v) deployment and training, and (vi) project management.

This effort estimation step to solve TD items was applied only in projects A, C and D. In projects B and E, the manager and technical team considered the estimation to be too complex. Project managers of projects C and D were also advised to use their preferred estimation method. Nevertheless, at the end, they followed a similar estimation method to the one used in Project-A (informal estimates per activity, supported by the project team whenever needed), they decided to even use the same generic activities.

IV. Data Analysis and Results

Hereafter we present our data analyses and results, organized by our case study research questions.

A. RQ1: Are usability debt items common in software development projects?

In order to answer this question, we used the amount of usability debt items found in the five projects, in comparison to the other types. The distribution of the 145 TD items per type for each project is shown in Table I. This table shows three columns for each project, concerning the number of items, the man hour effort estimation to address the items, and the effort distribution.

TABLE I
TD items (number and effort distribution) per type by project.

TD Types	Man-hour effort estimate														
	Company-1						Company-2			Company-3			Company-4		
	Project-A			Project-B			Project-C			Project-D			Project-E		
	Item	M/H	%	Item	M/H	%	Item	M/H	%	Item	M/H	%	Item	M/H	%
Architecture	2	1,680	58.8%	1	-	-				4	108	17.7%	1	-	-
Code	11	198	6.9%	5	-	-	4	382	13.7%	6	74	12.1%	2	-	-
Defect	2	32	1.1%	2	-	-	5	373	13.4%	11	175	28.7%	3	-	-
Design					-	-				6	27	4.4%	2	-	-
Doc.	3	292	10.2%		-	-				4	20	3.3%	1	-	-
Infrastructure					-	-	11	885	31.8%					-	-
Process					-	-	2	884	31.7%				1	-	-
Requirement	4	464	16.2%	10	-	-	2	90	3.2%	11	170	27.9%	7	-	-
Service				1	-	-								-	-
Test					-	-				1	5	0.8%		-	-
Usability	4	191	6.7%	5	-	-	3	173	6.2%	5	31	5.1%	3	-	-
Total	26	2,857	100.0%	24	-	-	27	2,787	100.0%	48	610	100.0%	20	-	-

While the data of different projects should not be directly aggregated for more sophisticated statistic purposes, the frequency of items per TD type helps to provide a descriptive overview, and is as follows: Requirements (23.5%), Code (19.3%), Defect (15.9%), Usability (13.8%), Infrastructure (7.6%), Architecture (5.5%), Design (5.5%), Documentation (5.5%), Process (2.1%), Service (0.7%), and Test (0.7%). Only four TD types occurred in all projects: Requirement, Code, Defect, and Usability.

Regarding usability debt, despite the lack of studies on the topic, TD items of this type were frequent and representative in our analyzed software development projects. In **Project-A**, 4 of the items (15.4%) were related to usability debt, being

the 2nd most frequent type (out of eleven analyzed types) in that project. In **Project-B**, 5 items (20.8%) were related to usability debt, also being the 2nd most frequent type. In **Project-C** there were 3 usability debt items (11.1%), ranking 4th. In **Project-D**, 5 (10.4%) of the TD items were related to usability debt, ranking 5th. Finally, in **Project-E**, 3 items (15%) were of usability debt, being the 2nd most frequent type.

Answering RQ1, the analysis confirmed that usability debt occurred frequently (ranging between 10.4% and 20.8% of the total TD items of the five projects), being also one of the only four types of debt that occurred in all projects.

B. RQ2: What types of usability debt usually occur in software development projects?

To answer this research question, after data collection, the usability items were classified according to Jacob Nielsen's usability heuristics [10] [11]. Such heuristics represent 10 general principles for interaction design, and were used in our work to analyze usability debt items: (1) Visibility of system status; (2) Match between system and the real world; (3) User control and freedom; (4) Consistency and standards; (5) Error prevention; (6) Recognition rather than recall; (7) Flexibility and efficiency of use; (8) Aesthetics and minimalist design; (9) Help users to recognize, diagnose, and recover from errors; and (10) Help and documentation.

The identification of the heuristics that were violated within the usability TD items was carried out by one of the authors with the support of an expert in human-computer interaction (PhD on the topic). Thereafter, the results were also presented to the managers of each of the five projects for validation. This step was important because in some cases the summary descriptions of the TD items did not provide much information to subsidize the classification. The result of the identified heuristics can be seen in Table II.

TABLE II
HEURISTICS VIOLATED BY USABILITY TD ITEMS.

#	NIELSEN HEURISTICS	Ticket	Project	Summary
1	1-Visibility of system status	1261	Project-D	Export to excel - Hold Cursor
2	2-Match between system and the real world	6635	Project-B	Correction in the description of the 2015 INR Proposals
3	3-User control and freedom	970	Project-D	Restoring Default Statement Ordering for Inconsistencies
4		11-SI	Project-E	List projects per unit / Unit manager view all proposals
5	5-Error Prevention	1293	Project-B	Enable saving of texts automatically
6		860	Project-D	Adjust filter drive click sensitivity [Fuse Parameters screen]
7		970	Project-D	Restoring Default Statement Ordering for Inconsistencies
8		965	Project-D	Disable SAVE button after project saving
9	6-Recognition rather than recall	891	Project-B	Proposal filter
10		821	Project-D	Slightly different TAGs (Add tag identifier)
11		13-SI	Project-E	Show when action was not taken without clicking details
12	7-Flexibility and efficiency of use	639	Project-B	Management Report - RNI
13		2666	Project-B	Automatic generation of bad debts
14		7035	Project-A	Allow Sort of Check Items in Different Topics
15		4510	Project-A	Link to bring PDF of APLIC
16		2956	Project-A	Charts (From charts allow the generation of graphs.
17		43	Project-C	Access to certain systems when the user is a Unit Manager
18		12-SI	Project-E	Register function from team member screen
19	8-Aesthetics and minimalist design	29	Project-C	Exchanging the SAHRA Report Template
20	9-Help users to recognize, diagnose, and recover from errors	2349	Project-A	Verification Item Message Correction in Technical Report Processing
21		190	Project-C	Most Exceptional Exception Pages

It is known that a given interaction problem may be violating more than one heuristic. In our case study, we observed that only one item (ticket # 970 - Project-D) violated more than one heuristic as presented in Table II.

The usability TD items observed in our analyzed projects regard violating eight (1, 2, 3, 5, 6, 7, 8, and 9) different (out of ten) heuristics. The items relate to deliberately postponed interaction problems, mainly concerning "7-Flexibility and efficiency of use", with 7 occurrences (33.3%), "5-Error prevention" with 4 occurrences (19.0%) and "Recognition rather than recall" with 3 occurrences (14.3%). To provide a better understanding on the usability TD items, a short description of the three heuristics that stood out by concentrating 66.6% of the selected items follows:

5-Error prevention. This heuristic, representing 19.0% of the usability TD item heuristic violations, states that the system should prevent mistakes and errors from occurring whenever possible [11]. The observed usability TD items regard not avoiding situations in which the system may lead to errors or is causing confusion due to the need for adjustments.

6-Recognition rather than recall. Representing 14.3% of the usability TD item heuristic violations, this heuristic states that the interface should clearly present the objects, actions and options, since the user should not need to "memorize". The user should not have to remember information from one part of the dialog to another [11]. Indeed, the TD items related to this category had in common a difficulty of the user in understanding the actions of the system.

7-Flexibility and efficiency of use: Representing 33.3% of the usability TD item heuristic violations, this was the most frequent one. It involves allowing users to customize frequent actions [11]. In fact, the TD items classified as violations of this heuristics were requests from experienced users requiring more effective usability alternatives for a systems functionality.

The answer that we can provide for RQ2, as our sample allowed observing instances violating eight out of ten of Nielsen's usability heuristics, is that different types of usability debt tend to occur during software development projects. The most common violated heuristics in our sample concerned "Flexibility and efficiency of use", followed by "Error prevention". It is noteworthy that these kinds of violations tend to have high negative impact on user satisfaction.

C. RQ3: What is the effort to resolve usability debt items in software development projects?

To answer this research question we used the man hour effort estimates provided by the project managers for resolving the TD items. As previously mentioned, only the project managers of projects A, C, and D provided estimates, totaling estimations for 101 TD items involving a total estimated effort of 6.254 man hours. The effort estimates to resolve the TD items for each TD type per project are shown in Table I.

When analyzing the usability TD item resolution effort to answer RQ3, we observed that, while being frequent and relevant (cf. analyses for RQ1 and RQ2), usability debt items typically require low resolution effort (6.7% for Project A, 6.2% for Project C, and 5.1% for Project D). Thus, from the perspective of TD management, paying this type of debt typically has a positive trade-off, eliminating a large number of items from the projects' TD portfolio with low effort while

also helping to improve the user experience for the systems under development.

V. Threats to validity

We next discuss threats to the validity of this research in relation to the four types suggested by [14].

Construct validity. The purpose of this multi-case study was to categorize usability TD items in terms of their frequency, types and resolution effort. The metric used for measuring effort, man hour, is widely used in software projects and was already in use in the selected projects. Furthermore, when managers and the technical team selected, classified and estimated the effort for the TD items, they were not informed that the focus of our research was on usability TD items.

Internal validity. This case study was anchored in information from real projects of companies that use the information about their tasks in an organized and methodical way. Effort estimates are subject to errors, which could also compromise internal validity. However, they were carried out by experienced professionals, following their traditional way of accomplishing this task. Moreover, estimate errors would likely affect the TD items of different types similarly.

External validity. The study involved five projects regarding systems that are in production and under active maintenance, developed by four different companies, with qualified IT professionals. Our sample of projects comprised different application domains, team sizes, development methods (e.g., two companies were mainly agile and two were plan-driven), and development technologies. Nevertheless, despite the rich diversity of settings, it is not possible to state that theoretical saturation has been achieved. Hence, the results can not be generalized and further replications of the study are needed.

Reliability. Data collection was carefully conducted following our predefined procedure (*cf.* Section II.C) and appropriately documented (*cf.* Section III). All TD item selections, classifications and estimates were peer reviewed by experts and validated with project team representatives. Hence, we triangulated the collected data by conducting validations directly interacting with the project teams. Therefore, we believe that similar results would be produced if the data collection procedures were applied in these projects by other researchers.

VI. Concluding Remarks

The purpose of this paper was to characterize the occurrence of usability debt in the context of software development. We performed a multi-case study, analyzing TD items from five software projects of four different companies with a diversity of settings. Our main findings indicate that, in the analyzed projects, usability debt: (i) is frequent; (ii) regards violating several relevant user interaction principles, which may negatively affect user satisfaction; and (iii) requires low resolution effort.

Indeed, usability debt occurred in all projects. Usability TD items ranged from 10.4% do 20.8% of all TD items within the projects. We observed that our sample of usability TD items concerned violations of eight different (out the ten possible) Nielsen usability heuristics. The most common violated heuristic concerned “Flexibility and efficiency of use” and “Error prevention”. These kinds of violations, by their definition, tend to have a negative effect on user satisfaction. Finally, we observed that usability TD items typically require a low resolution effort, ranging from 5.1% to 6.7% of the total estimated TD item resolution effort within the analyzed projects. Based on these results, we put forward that practitioners should consider this type of TD in their TD management strategies. We recommend to identify and address usability debt items, eliminating them from the project TD portfolio, potentially generating a positive impact on the user.

Being the first study focusing on the occurrence of usability debt in industrial practice, we believe that it contributes towards an initial understanding of the phenomenon, opening avenues of research for future work in this area.

References

[1] N. S. Alves, T. S. Mendes, M. G. de Mendona, R. O. Spınola, F. Shull, C. Seaman, ”Identification and management of technical debt: A systematic mapping study.”, Information and Software Technology, vol. 70, pp. 100-121, 2016.

[2] A. Ampatzoglou et al., ”The Financial Aspect of Managing Technical Debt: A Systematic Literature Review”, Information and Software Technology, pp. 52-73, 2015.

[3] W. Cunningham , ”The WyCash portfolio management system.”, in Addendum to the proceedings on Object-oriented programming systems, languages, and applications (Addendum) (OOPSLA ’92), 1992, pp. 29-30.

[4] D. Falessi, P. Kruchten, P. Avgeriou, Introduction to the special issue on technical debt in software systems, Journal of Systems and Software, vol. 120 issue C, pp. 154155, 2016.

[5] C. Fernandez-Sanchez, J. Garbajosa, A. Yagüe, J. Perez,” Identification and analysis of the elements required to manage technical debt by means of a systematic mapping study,” Journal of Systems and Software, vol. 124, pp. 2238, 2017.

[6] Y. Guo, C. Seaman, F. Q. B. da Silva, ”Costs and obstacles encountered in technical debt management a case study, Journal of Systems and Software, vol. 120, pp. 156169, 2016.

[7] P. Kruchten, R. L. Nord, I. Ozkaya, ”Technical debt: From metaphor to theory and practice”, IEEE Software, vol. 29, no. 6, pp. 18-21, 2012.

[8] M. M. Lehman, Feedback in the software evolution process,, Information and Software Technology, vol. 38, Issue 11, pp. 681-686, 1996.

[9] Z. Li, P. Avgeriou, P. Liang, ”A Systematic Mapping Study on Technical Debt and Its Management”, Journal of Systems and Software, vol. 101, pp. 193-220, 2015.

[10] R. Molich, J. Nielsen, Improving a human-computer dialogue,, Communications of the ACM, vol. 33 , no. 3, pp. 338-348, 1990.

[11] J. Nielsen, 10 Usability Heuristics for User Interface Design, [online] Available: http://www.nngroup.com/articles/ten-usability-heuristics.

[12] A. Potdar, E. Shihab, An Exploratory Study on Self-Admitted Technical Debt,” in Proceedings of the 2014 IEEE International Conference on Software Maintenance and Evolution (ICSME ’14), 2014, pp. 91-100.

[13] N. Rios, M.G. Mendona, and R.O. Spinola, A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners, Information and Software Technology, vol 102, pp. 117-145, 2018.

[14] P. Runeson, M. Host, A. Rainer, and B. Regnell, Case Study Research in Software Engineering- Guidelines and Examples, John Wiley & Sons, 2012.

[15] C. Seaman, and Y. Guo, Chapter 2 - measuring and monitoring technical debt, Advances in Computers, vol. 82, pp. 2546, 2011.

[16] E. Tom, A. Aurum, R. Vidgen, ”An Exploration of Technical Debt”, Journal of Systems & Software, vol. 86, no. 6, pp. 1498-1516, 2013.

[17] N. Zazworka, R. O. Spınola, A. Vetro, F. Shull, and C. Seaman, A case study on effectively identifying technical debt, in Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering (EASE 13), 2013, pp. 42-47.