Debt Stories: Capturing Social and Technical Debt in the Industry

Nicolas Riquet
NADI, University of Namur
Namur, Belgium
nicolas.riquet@unamur.be

Xavier Devroey
NADI, University of Namur
Namur, Belgium
xavier.devroey@unamur.be

Benoît Vanderose
NADI, University of Namur
Namur, Belgium
benoit.vanderose@unamur.be

ABSTRACT

In today's organizations, software is mission-critical. However, the legacy of past decisions can make tasks related to artifacts increasingly inefficient or risky, creating debt. While most researchers and practitioners mainly focus on technical debt, some have investigated its social dimensions, known as social debt. We argue that organizations developing software need to tackle debt holistically, as it is intrinsically a socio-technical issue. In this short paper, we rely on a definition of socio-technical debt based on the existing literature to define Debt Stories: a tool based on the User Story format, that can help capture debt elements directly from the stakeholders involved in software development. A debt story includes information about the role of the stakeholder in the development process, the social or technical context, and the impact of the debt element on the different tasks performed by the stakeholder. We provide a first empirical evaluation of the usage of Debt Stories in an industrial context, demonstrating the relevance of Debt Stories to express and communicate socio-technical debt.

CCS CONCEPTS

• **Software and its engineering** → *Agile software development*; **Software maintenance tools**; • **Social and professional topics** → **Software maintenance**.

KEYWORDS

socio-technical debt, debt stories, empirical software engineering

ACM Reference Format:
Nicolas Riquet, Xavier Devroey, and Benoît Vanderose. 2024. Debt Stories: Capturing Social and Technical Debt in the Industry. In *International Conference on Technical Debt (TechDebt '24), April 14–15, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3644384.3644473

1 INTRODUCTION

Software is critical to the efficient operations, performance and productivity of a vast majority of organizations [13, 38]. As a result, executives increasingly integrate technology aspects to their organization's strategy to preserve their competitive advantage [22]. However, developing and maintaining applications that can endure the passage of time is not easy: new requirements are expressed, change requests are made, bugs are found, and an application must evolve to remain relevant to the business. And implementation choices can have a significant impact on the quality, and development and maintenance time. For instance, shortcuts taken to implement the changes in the required time frame or issues left for later fixes can require heavy refactoring afterwards. If such problems accumulate over time, they can become a liability as they make it increasingly harder to bring changes to the concerned applications in a timely, cost-effective, and quality-oriented way. This phenomenon is referred to by both researchers and practitioners as technical debt, a metaphor first used by Ward Cunningham [11, 12]: "*the longer you sit on technical debt of any kind, the more time-consuming and expensive it will be to fix.*"

Technical debt is a widely used concept nowadays but there is no real consensus on what exactly it encompasses [45] and many different definitions have been proposed [2, 3, 7, 20, 23, 26, 39, 48]. The term *technology debt* is sometimes used in management circles as a synonym of technical debt or as a somewhat broader concept, and its impact on the performance of organizations is quite well documented [18, 28–30]. More recently, Tamburri et al. [42–44] focused on the social aspects of debt in software engineering and defined social debt as "*the unforeseen project cost connected to a 'sub-optimal' development community*" [42]. It may comprise elements such as community smells [21, 24, 41, 44], communication networks [15], and culture [15, 17]. Social debt researchers also use the term *socio-technical* when discussing phenomenons or dynamics that involve both social and technical dimensions, or when one of these dimensions affects the other. For instance, *Conway's Law* [10] (i.e., "*Any organization that designs a system will produce a design whose structure is a copy of the organization's communication structure.*") shows the impact of an organization's communication structures on the design of systems. To the best of our knowledge, there is, however, no common definition of *socio-technical* debt. For organizations, debt is a source of technical, social, and organizational risks that must be identified and addressed [5]. They need to identify and tackle both technical debt in the artifacts they produce and social debt in their organizational structures and processes. This is not trivial as it requires first to be able to identify debt elements.

In this work, we aim for an inclusive notion of debt to focus on capturing social and technical debt for software engineering, denoted *Socio-Technical Debt* (STeD), directly from the stakeholders involved in the software development and maintenance lifecycle. For that, we rely on the use of *debt stories*, a tool we created based on the User Story widely used in Agile development to describe a user's need or request [4, 14] and that helps developers understand and document the context and what the software does [34]. Our contributions include: (i) a definition of Socio-Technical Debt; (ii) a tool for identifying and discussing debt elements in organizations developing software in the form of debt stories; (iii) a first empirical evaluation of the usage of debt stories in an industrial context; and (iv) a replication package [36] containing the necessary material to replicate our evaluation.

TechDebt '24, April 14–15, 2024, Lisbon, Portugal

Industrial context. Our evaluation took place in a Belgian French-speaking organization counting more than 5,000 employees.[1] It has 148 development professionals occupying 16 different roles and distributed across 19 teams.The organization maintains over 140 applications for a total of over 11 millions lines of code and actively develops new ones. The development teams implement Scrum with some adjustments in order to be compatible with higher-level Prince2 project management. The organization is facing a significant level of technical debt and has initiated efforts to try to reduce it. This work has shown that social aspects are also at play and that it is necessary to approach the problem holistically.

2 CAPTURING SOCIO-TECHNICAL DEBT

We base our definition of debt for software engineering on the various existing definitions of social and technical debt we found in the literature [2, 3, 7, 15, 17, 18, 20, 23, 26, 28–30, 39, 41–45, 48].

Definition 2.1 (Socio-Technical Debt (STeD)). Socio-Technical Debt refers to a phenomenon that occurs when past social and/or technical decisions lead to inefficiencies or increased risks in designing, building, maintaining, managing, or using software engineering artifacts. This, in turn, hampers an organization's ability to create value, achieve strategic goals, or pursue the best course of action. Socio-Technical Debt has the following characteristics: (1) it is cumulative in nature: the longer it stays, the more it costs in the long run; (2) it requires some effort to eliminate. Significant time and/or money must be invested to pay it off; and (3) it is inevitable in most software engineering projects as trade-offs are always being made, either knowingly or unknowingly. A certain level of Socio-Technical Debt can always be deemed sustainable, and the return on investment of paying debt items must be carefully evaluated.

Identifying and tracking Socio-Technical Debt is not trivial. Existing approaches have focused on source code analysis, and mining software repositories and issue trackers to identify antipatterns like code smells and bad programming practices [19, 31, 37, 40, 46, 47], community [21, 24, 41, 44] or architectural smells [8, 25, 49], loss of code knowledge and ownership [6, 33, 35], etc. Tools like SonarQube are used in the industry and can help software engineering teams to manage their debts. The majority of those methods rely on heuristics and focus on specific issues to automatically identify debt elements. Unlike automated approaches, we rely directly on stakeholders involved in the software engineering process to elicit debt elements, allowing a wider range of potential issues.

Debt story. To identify and discuss socio-technical debt elements, we define the debt story format. Similarly to User Stories, widely used in Agile development [4, 14], debt stories allow one to express and document the context of a debt element. The most commonly used format for writing a User Story is "*As a **[who]** I want to **[what]** so that **[why]**,*" also known as the Connextra format [1], but there exist several variations [9, 27]. In our case, a debt story includes information about the role of the stakeholder in the development process, the social or technical context, and the the impact of the debt element on the different tasks performed by the stakeholder.

[1]For legal and public relations reasons the organization wishes to remain anonymous (the authors can be contacted for more details on the organization).

Definition 2.2 (debt story). A debt story is a tool to identify and discuss socio-technical debt elements. It is formatted as follows:

> As a(n) **[actor role]** of **[social or technical context]**
> I find that it is increasingly **[impact type]** to **[task]**
> because **[debt item]**.

Here are two debt stories used as examples during the second part of our evaluation: (i) *As a back-end developer of Application A, I find that it is increasingly time-consuming to maintain the document generation module because the code has become too complex.* (ii) *As a(n) developer of the A team, I find that it is increasingly frustrating to integrate with Application C because the B team does not take our needs into consideration and keeps breaking compatibility.*

3 PRELIMINARY EVALUATION

We tested the potential impact of our idea by performing an empirical case study within our industrial context. We organised workshops with ten development experts and two managers to answer the following research questions:

- **RQ1** How do development experts and managers perceive the usefulness of *debt stories* in *expressing* debt elements?
- **RQ2** How do development experts and managers perceive the usefulness of *debt stories* in *communicating* debt elements?

While **RQ1** focuses on the expressivity of debt stories, **RQ2** focuses on its adequacy to communicate debt elements to other development experts and managers. In a nutshell, we presented to the participants the suggested definition, our debt story format and asked them to use a debt story edition tool to communicate debt elements affecting their work. We then asked the participants to complete a questionnaire survey to share their opinion on the concepts and tools proposed.

Organization of the workshops. Three identical workshops were arranged: development experts were invited to either workshop 1 or 2 and managers to workshop 3. This categorization based on job roles was implemented to prevent development experts from self-censoring in the presence of managers. Development teams were notified within the organization through a message on *Microsoft Teams* explaining: (1) the goals of the study and how these were aligned with the needs of the organization; (2) when and how the study would take place; (3) that volunteers simply had to send a private *Microsoft Teams* message or an email to the authors to participate; (4) that the participants' data would be anonymized. This message was sent three times over a period of two weeks in order to make sure enough people saw it. In total, four development experts (one junior analyst, one front-end developer, one security expert, and one lead developer) participated to the first workshop, six (one junior developer, two senior developers, one Sharepoint developer, one web designer, and one team lead) participated to the second workshop, and two managers, including a director, participated to the third workshop.

Figure 1 presents an overview of the course of the workshops, each one lasting between 1:30 and 2:00 hours. (1) The first author, as workshop facilitator, presents the objectives of the overall research project, the organization's motivations for participating in it, and the scope of the study. Then (2) he offers and explains the definition of Socio-Technical Debt (see definition 2.1), reminds the participants of the objective and format of the User Story, and presents the

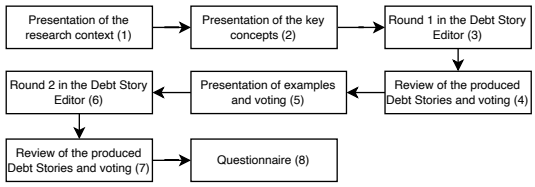

Figure 1: Workshop organization overview

format of the debt story (see definition 2.2). (3) The facilitator gives a demo of the *debt story editor* implemented as a spreadsheet and asks the participants to produce debt stories from scratch. (4) He presents the upvote sheet, used to vote for debt stories produced by others that describe a situation affecting the participant's work as well. Then the facilitator reads aloud the debt stories produced during round 1, reminding each time that people who had not created the debt story could vote for it by adding its identifier to their upvote sheet.

In the second part of the workshop, in step (5) , the facilitator starts by showing and reading aloud seven examples of debt stories created by the authors and, as previously, reminds the participants to vote for examples affecting their work. (6) Then the facilitator asks participants to open the second instance of the *debt story editor* containing the seven examples and produce debt stories. Participants did not have any specific instructions and could copy debt stories from previous rounds if they wanted. (7) Then, again, the facilitator reads aloud the debt stories for potential upvoting by the participants. (8) Finally, the facilitator asks the participants to respond to the questionnaire described hereafter.

Questionnaire. The questionnaire was divided into four groups of questions: (Part I) five questions about the participant's professional experience, (Part II) 10 closed-ended questions about their opinion on Socio-Technical Debt, (Part III) 15 closed-ended questions about their opinion on the debt story tool based on Likert-type scales, and (Part IV) six open-ended questions. The questionnaire is available in our replication package [36].

Data preparation and analysis. This study relies on both quantitative and qualitative data. Qualitative data was gathered from participants' answers to open-ended questions and the debt stories they created.[2] Quantitative data are derived from questionnaires by assigning values on a scale of 1 to 5 to Likert-type responses. As we used five-point scales, we adopted a neutral value of 3 for the *agreement*, *frequency*, and *clarity* scales and classify responses in broader categories: > 3 (positive rating), $= 3$ (neutral rating), and < 3 (negative rating). These broader categories were also employed for the other (non Likert-type) scales, when applicable, to further classify our participants: e.g., employee categories, such as *junior*, *medior*, and *senior* profiles, as defined by the organization. Additionally, we summed the number of upvotes received by each debt story to gain further insights.

4 RESULTS

Responses show that all 12 participants (100%) found the definition of Socio-Technical Debt (STeD) to be clear or perfectly clear. Eleven (91.7%) agreed or strongly agreed that STeD elements made their tasks less efficient and 10 (83.3%) agreed or strongly agreed that STeD elements affecting their work posed risks to the organization. Moreover, 10 (83.3%) reported that STeD elements affected their work often or very often, illustrating the importance of Socio-Technical Debt in our industrial context. However, only six participants (50%) reported that they often or very often discussed STeD elements affecting their tasks with their immediate colleagues, and only three (25%) said they discussed the subject often or very often with their manager or supervisor. Additionally, all 12 participants (100%) agreed or strongly agreed that they were familiar with the format of User Stories traditionally used in agile development to describe a user's need or request.

RQ1 (*Expressing* STeD). Eleven participants (91.7%) agreed or strongly agreed that the format proposed for the Debt Story allowed them to express STeD elements that affected their tasks, and all 12 (100%) agreed or strongly agreed that it provided them with enough flexibility to cover both technical and social debt elements affecting their work. Ten (83.3%) agreed or strongly agreed that the fact that the Debt Story was inspired by the User Story made it easier for people with experience in agile development to adopt the tool. Eight participants (66.7%) agreed or strongly agreed that they found it easy to formulate Debt Stories during round 1 of the workshops (before having seen any examples) and all 12 participants (100%) agreed or strongly agreed that they found it easy to formulate Debt Stories during round 2 after having seen some examples, confirming the relevance of our format in Def. 2.2 in our industrial context.

A total of 97 Debt Stories were generated during the workshops. One of these was eliminated because it was formulated as a joke. Out of the remaining 96, 69 received at least one upvote. To assess their alignment with Def. 2.1, each of the three first authors individually evaluated each Debt Story and discussed to reach a consensus in case of disagreement. The Fleiss' kappa [16] inter-rater agreement is 0.141 ($p - value = 0.0166$), indicating *slight agreement*. This number can be explained by the background of the three authors. Only the first author is part of our industrial context, and, as such, has a deeper knowledge and understanding of the collected debt stories. In total, 69 Debt Stories (71.9%) were considered consistent with the proposed definition for STeD, while the others primarily addressed organizational issues that *could* lead to STeD but do not concern an identified software engineering artifact. In our future work, we plan to further analyse the collected debt stories to identify the different STeD elements discussed by the participants, and replicate our evaluation in a different context to strengthen our conclusions.

RQ2 (*Communicating* STeD). Eleven participants (91.7%) found that the Debt Stories from other people are clear or perfectly clear to them. Eleven (91.7%) agreed or strongly agreed that the Debt Story is a suitable tool for discussing socio-technical debt elements, and 11 (91.7%) agreed or strongly agreed that the Debt Story is a tool that encourages bringing STeD issues affecting their work to the attention of their hierarchy. Ten participants (83.3%) agreed or strongly agreed that using Debt Stories would bring value to the organization, and eight (66.7%) agreed or strongly agreed that there would be added value in integrating the use of Debt Stories into their tools (e.g., Azure DevOps boards). Those results suggest that

[2]All answers were translated into English to ensure replicability of the data analysis. Translation and deeper analysis of the debt stories is part of our future work.

Debt Stories can indeed be used as a basis for communicating STeD elements in our industrial context.

5 DISCUSSION

Our results indicate that debt stories can help both development experts and managers to express and communicate debt elements in our industrial context. Although confirming those results in another context is part of our future work, we see that there are no differences in the responses between development experts and managers. Managers did report, however, that they discussed debt elements with their immediate colleagues or their own managers less frequently. This could be expected, as their (manager) colleagues are responsible for unrelated activities, and their own managers are less involved in operational matters.

Expressing STeD. Responses to the open-ended questions indicate that 8 participants (66.7%) did not find that there were elements that made their tasks less efficient or riskier and that were not covered by the proposed definition of socio-technical debt. Some participants mentioned that personal behavioral issues, higher-level process aspects, and lack of strategy were potentially not covered. Although important, we believe that these elements are broader organizational issues that are out of the scope of Def. 2.1 when they do not involve software engineering artifacts. Such elements were observed in the collected debt stories but were hard to identify without a deeper knowledge of our industrial context, also explaining the *slight* inter-rater agreement observed for RQ1.

When participants were asked open-ended questions about the difficulties they encountered in formulating Debt Stories, one participant mentioned that several impact types could be used to describe a given issue. Indeed, a time-consuming debt element incurs higher costs and represents a budget risk. However, we do not perceive this as an issue, as the primary purpose of the tool is to identify and discuss issues in a semi-structured format. The tool is flexible and allows for conveying a broad spectrum of impact types. Debt Stories can and should be further elaborated upon during later discussions. Similarly, some participants reported that it was not always easy to choose the right role or context, as several could be suitable. Our advice is to choose the first one that comes to mind and discuss this choice in a later phase.

Communicating STeD. Most difficulties are related to the fact that some debt stories were too specific to a role or context, making it challenging for participants to relate. Some participants also mentioned that certain debt stories produced by other participants were too long or complex. The same reasons were provided by most people when asked why they did not upvote a particular Debt Story. When asked about modifications they would like to make to Def. 2.2, half of the participants reported that they would not change anything, while the others mentioned issues with some debt stories that are too long or complicated, or for which the impact types or roles are questionable. Although these remarks are not directly related to formatting issues (as confirmed by the answers to the closed-ended questions), they do point out the importance of adhering to shared guidelines and best practices when creating and using debt stories. This would ensure maximum clarity and relevance across different roles and contexts.

Managing STeD. One participant expressed the need to involve people from other departments (such as IT Infrastructure, User Support, etc.) to effectively and efficiently manage debt. This is a valuable insight, which we will address in our future work to define a methodology and tools to help characterize socio-technical debt items and determine appropriate mitigation actions. We also plan to study the relationship between socio-technical debt and DevOps concepts and practices, among which those related to the responsibilities of IT operations experts.

6 THREATS TO VALIDITY

Internal validity. To prevent priming during the creation of debt stories, we ensured that examples were presented only after the initial round in the Debt Story editor tool (step 5 in Figure 1). Also, to ensure that participants did not upvote their own debt stories, we employed debt story identifiers that included the participant's number. Regarding the questionnaire, we used distinct formatting to prevent any confusion between the terms "Debt Story" and "User Story": the former was consistently written in bold font, while the latter was underlined. The translation of the content and data into English was carried out to the best of our abilities, and we sought assistance from ChatGPT [32], a large language model-based chatbot, when we were unsure about the most idiomatic way of translating some sentences. We plan to apply the same approach for translating the debt stories collected during the evaluation.

External validity. Our workshops involved only 12 participants, all from the same organization, which limits the generalizability of our results (a common threat when performing empirical evaluation in an industrial context). However, this threat is mitigated by several elements: the organization employs standard tools and methodologies, making it a representative example of large enterprises that develop software to support complex business operations. It has been engaged in application development for decades, which has exposed it to the challenges of adapting to evolving business needs, including the creation of new applications and substantial maintenance efforts for existing ones. Regarding the selection of the participants, we believe it is representative of the various teams, roles, and career experience levels within the development department. Replicating our evaluation in a different industrial context to further address external validity is part of our future work.

Replicability. The replication package [36] includes the following (both in English and in French): the Debt Story editor template and instances for rounds 1 and 2, the upvote sheet, the questionnaire and the responses data. This case study was conducted in French and all English translations were done for the purpose of reproducibility.

7 CONCLUSION

In this short paper, we presented debt stories, a tool to help stakeholders involved in the software development and maintenance lifecycle to express and communicate STeD elements. Our initial results confirm that Socio-Technical Debt and debt stories would indeed help software engineering practitioners from our industrial context expressing and communicating debt elements.

In our future work, we plan to replicate our workshop in other industrial context and analyse the debt stories produced more in-depth to identify which actor roles, social or technical context,

impact type, and debt elements are expressed. Based on this analysis we will propose methods and refined tools to assist teams and their managers in characterizing the identified debt elements, defining the possible mitigating actions, and their prioritization. We also intend to investigate how the various aspects of Socio-Technical Debt intersect with common DevOps concepts such as waste and how the implementation of DevOps practices and tools can help reduce the overall debt level and the production of new debt. The overall goal of our research project is to propose a management framework for defining, identifying, characterizing, communicating, and managing Socio-Technical Debt.

REFERENCES

[1] Agile Alliance. 2020. User Story Template. https://www.agilealliance.org/glossary/user-story-template. Accessed: 2023-08-07.

[2] Areti Ampatzoglou, Apostolos Ampatzoglou, Paris Avgeriou, and Alexander Chatzigeorgiou. 2015. Establishing a framework for managing interest in technical debt. In *BMSD '15*.

[3] Paris Avgeriou, Philippe Kruchten, Ipek Ozkaya, and Carolyn Seaman. 2016. Managing technical debt in software engineering (dagstuhl seminar 16162). In *Dagstuhl reports*, Vol. 6. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[4] Kent Beck. 2000. *Extreme programming explained: embrace change*. addison-wesley professional.

[5] Terese Besker, Antonio Martini, and Jan Bosch. 2017. Time to Pay Up: Technical Debt from a Software Quality Perspective. In *CIbSE*. 235–248.

[6] Christian Bird, Nachiappan Nagappan, Brendan Murphy, Harald Gall, and Premkumar Devanbu. 2011. Don't Touch My Code! Examining the Effects of Ownership on Software Quality. In *ESEC/FSE '11* (Szeged, Hungary). ACM, 4–14.

[7] Nanette Brown, Yuanfang Cai, Yuepu Guo, Rick Kazman, Miryung Kim, Philippe Kruchten, Erin Lim, Alan MacCormack, Robert Nord, Ipek Ozkaya, et al. 2010. Managing technical debt in software-reliant systems. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*. 47–52.

[8] Alexandra-Maria Chaniotaki and Tushar Sharma. 2021. Architecture Smells and Pareto Principle: A Preliminary Empirical Exploration. In *MSR '21*. 190–194.

[9] Mike Cohn. 2004. *User stories applied: For agile software development*. Addison-Wesley Professional.

[10] Melvin E Conway. 1968. How do committees invent. *Datamation* 14, 4 (1968), 28–31.

[11] Ward Cunningham. 1992. The WyCash portfolio management system. *ACM SIGPLAN OOPS Messenger* 4, 2 (1992), 29–30.

[12] Cunningham, Ward. 2009. Debt Metaphor. https://www.youtube.com/watch?v=pqeJFYwnkjE. Accessed: 2023-03-27.

[13] Jason Dedrick, Vijay Gurbaxani, and Kenneth L Kraemer. 2003. Information technology and economic performance: A critical review of the empirical evidence. *ACM Computing Surveys (CSUR)* 35, 1 (2003), 1–28.

[14] Sonja Dimitrijević, Jelena Jovanović, and Vladan Devedžić. 2015. A comparative study of software tools for user story management. *Information and Software Technology* 57 (2015), 352–368.

[15] Neil Ernst, Rick Kazman, and Julien Delange. 2021. *Technical Debt in Practice*. The MIT Press.

[16] Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin* 76, 5 (Nov. 1971), 378–382.

[17] Nicole Forsgren, Jez Humble, and Gene Kim. 2018. *Accelerate: The science of lean software and devops: Building and scaling high performing technology organizations*. IT Revolution.

[18] Gartner. 2020. Manage Technology Debt to Create Technology Wealth. https://www.gartner.com/en/documents/3989188. Accessed: 2023-03-21.

[19] Tracy Hall, Min Zhang, David Bowes, and Yi Sun. 2014. Some Code Smells Have a Significant but Small Effect on Faults. *ACM Transactions on Software Engineering and Methodology* 23, 4 (2014), 1–39.

[20] Johannes Holvitie, Sherlock A Licorish, Rodrigo O Spínola, Sami Hyrynsalmi, Stephen G MacDonell, Thiago S Mendes, Jim Buchan, and Ville Leppänen. 2018. Technical debt and agile software development practices and processes: An industry practitioner survey. *Information and Software Technology* 96 (2018), 141–160.

[21] Zijie Huang, Zhiqing Shao, Guisheng Fan, Jianhua Gao, Ziyi Zhou, Kang Yang, and Xingguang Yang. 2021. Predicting Community Smells' Occurrence on Individual Developers by Sentiments. In *ICPC '21*. 230–241.

[22] Jeff Immelt. 2015. GE CEO Jeff Immelt: Let's Finally End the Debate over Whether We Are in a Tech Bubble. *Business Insider* 9 (2015).

[23] Clemente Izurieta and James M Bieman. 2013. A multiple case study of design pattern decay, grime, and rot in evolving software systems. *Software Quality Journal* 21 (2013), 289–323.

[24] Stefano Lambiase, Gemma Catolino, Damian A. Tamburri, Alexander Serebrenik, Fabio Palomba, and Filomena Ferrucci. 2022. Good Fences Make Good Neighbours? On the Impact of Cultural and Geographical Dispersion on Community Smells. In *ICSE-SEIS '22*. ACM, 67–78.

[25] Ruiyin Li, Peng Liang, Mohamed Soliman, and Paris Avgeriou. 2022. Understanding software architecture erosion: A systematic mapping study. *Journal of Software: Evolution and Process* 34, 3 (March 2022), e2423.

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

[27] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn EM van der Werf, and Sjaak Brinkkemper. 2016. The use and effectiveness of user stories in practice. In *REFSQ '16* (Gothenburg, Sweden). Springer, 205–222.

[28] Johan Magnusson and Bendik Bygstad. 2014. Technology debt: Toward a new theory of technology heritage. In *ECIS '14*. 9–11.

[29] Johan Magnusson, Carlos Juiz, Beatriz Gómez, and Belén Bermejo. 2018. Governing technology debt: beyond technical debt. In *TecDebt '18*. 76–84.

[30] McKinsey. 2022. Demystifying digital dark matter: A new standard to tame technical debt. https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/demystifying-digital-dark-matter-a-new-standard-to-tame-technical-debt. Accessed: 2023-04-25.

[31] Steffen M. Olbrich, Daniela S. Cruzes, and Dag I.K. Sjøøberg. 2010. Are all code smells harmful? A study of God Classes and Brain Classes in the evolution of three open source systems. In *ICSM '10*.

[32] OpenAI. 2022. ChatGPT. https://openai.com/chatgpt. Accessed: 2023-08-07.

[33] Foyzur Rahman and Premkumar Devanbu. 2011. Ownership, Experience and Defects: A Fine-Grained Study of Authorship. In *ICSE '11* (Waikiki, Honolulu, HI, USA). ACM, 491–500.

[34] Paul Ralph. 2015. The sensemaking-coevolution-implementation theory of software design. *Science of Computer Programming* 101 (2015), 21–41.

[35] Nicolas Riquet, Xavier Devroey, and Benoît Vanderose. 2022. GitDelver Enterprise Dataset (GDED): An Industrial Closed-source Dataset for Socio-Technical Research. In *MSR '22*. Pittsburgh, PA, USA.

[36] Nicolas Riquet, Xavier Devroey, and Benoît Vanderose. 2024. https://zenodo.org/doi/10.5281/zenodo.10518270 Replication package.

[37] José Amancio M. Santos, João B. Rocha-Junior, Luciana Carla Lins Prates, Rogeres Santos do Nascimento, Mydiã Falcão Freitas, and Manoel Gomes de Mendonça. 2018. A systematic review on the code smell effect. *Journal of Systems and Software* 144, March (2018), 450–477.

[38] Petra Schubert and Uwe Leimstoll. 2007. Importance and use of information technology in small and medium-sized companies. *Electronic Markets* 17, 1 (2007), 38–55.

[39] Carolyn Seaman and Yuepu Guo. 2011. Measuring and monitoring technical debt. In *Advances in Computers*. Vol. 82. Elsevier, 25–46.

[40] Dag I.K. Sjoberg, Aiko Yamashita, Bente C.D. Anda, Audris Mockus, and Tore Dyba. 2013. Quantifying the effect of code smells on maintenance effort. *IEEE Transactions on Software Engineering* 39, 8 (2013), 1144–1156. ISBN: 0098-5589.

[41] Damian A Tamburri, Rick Kazman, and Hamed Fahimi. 2016. The architect's role in community shepherding. *IEEE Software* 33, 6 (2016), 70–79.

[42] Damian A Tamburri, Philippe Kruchten, Patricia Lago, and Hans van Vliet. 2013. What is social debt in software engineering?. In *CHASE '13*. IEEE, 93–96.

[43] Damian A Tamburri, Philippe Kruchten, Patricia Lago, and Hans van Vliet. 2015. Social debt in software engineering: insights from industry. *Journal of Internet Services and Applications* 6 (2015), 1–17.

[44] Damian A Tamburri, Fabio Palomba, and Rick Kazman. 2019. Exploring community smells in open-source: An automated approach. *IEEE Transactions on software Engineering* 47, 3 (2019), 630–652.

[45] Edith Tom, Aybüke Aurum, and Richard Vidgen. 2013. An exploration of technical debt. *Journal of Systems and Software* 86, 6 (2013), 1498–1516.

[46] Michele Tufano, Fabio Palomba, Gabriele Bavota, Rocco Oliveto, Massimiliano Di Penta, Andrea De Lucia, and Denys Poshyvanyk. 2017. When and Why Your Code Starts to Smell Bad (and Whether the Smells Go Away). *IEEE Transactions on Software Engineering* 43, 11 (Nov. 2017), 1063–1088.

[47] Aiko Yamashita and Leon Moonen. 2013. To what extent can maintenance problems be predicted by code smell detection? -An empirical study. *Information and Software Technology* 55, 12 (2013), 2223–2242.

[48] Nico Zazworka, Michele A. Shaw, Forrest Shull, and Carolyn Seaman. 2011. Investigating the Impact of Design Debt on Software Quality. In *Proceedings of the 2nd Workshop on Managing Technical Debt* (Waikiki, Honolulu, HI, USA) *(MTD '11)*. ACM, 17–23.

[49] Chenxing Zhong, Huang Huang, He Zhang, and Shanshan Li. 2022. Impacts, causes, and solutions of architectural smells in microservices: An industrial investigation. *Software: Practice and Experience* 52, 12 (Dec. 2022), 2574–2597.