What is Social Debt in Software Engineering

Tamburri, D.A.; Kruchten, P.; Lago, P.; van Vliet, H.

published in
Proceedings 6th Inernational Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2013)
2013

document version
Publisher's PDF, also known as Version of record

What Is Social Debt in Software Engineering?

Damian A. Tamburri*, Philippe Kruchten†, Patricia Lago*, and Hans van Vliet*
*Department of Computer Science, VU University Amsterdam, The Netherlands, [d.a.tamburri, p.lago, j.c.van.vliet]@vu.nl
†Electrical and Computer Engineering, University of British Columbia, Vancouver, BC, Canada, pbk@ece.ubc.ca

***Abstract*—"Social debt" in software engineering informally refers to unforeseen project cost connected to a "suboptimal" development community. The causes of suboptimal development communities can be many, ranging from global distance to organisational barriers to wrong or uninformed socio-technical decisions (i.e., decisions that influence both social and technical aspects of software development). Much like technical debt, social debt impacts heavily on software development success. We argue that, to ensure quality software engineering, practitioners should be provided with mechanisms to detect and manage the social debt connected to their development communities. This paper defines and elaborates on social debt, pointing out relevant research paths. We illustrate social debt by comparison with technical debt and discuss common real-life scenarios that exhibit "sub-optimal" development communities.**

***Index Terms*—Social Debt; Software Engineering; Human Factors in Software Engineering; Global Software Engineering; Technical Debt; Software Architecture; Social Communities; Social Structure; Social Networks;**

I. Introduction

Software engineering success is increasingly dependent on the well-being of developers' communities [1]. In previous work [2], [3] we found many decisions influencing community well-being. For example, changing the structure of the development community (e.g., through outsourcing), changing the development process (e.g., by adopting agile methods), leveraging on global collaboration (e.g., by striking a balance between formal and informal communication across global sites) are all socio-technical decisions, i.e., social and technical at the same time, that influence the state and welfare of developing communities and their members [4]. The social connotation of these decisions, changes the way people interact with others in the development community (e.g., collaboration between closed- and open-source is subject to formal filtering protocols) [2]. The technical connotation of these decisions, changes the way in which development tasks are worked out (e.g., in agile methods a "pull" task-allocation is used, as opposed to classic "push"). Wrong or uninformed socio-technical decisions cause additional cost on software projects and the development community around them. This cost is not immediately visible and its resolution is often postponed. Also, being connected to people, the cost increases as more projects are worked out by the same, "faulty" community.

This extra cost is conceptually similar to technical debt [5]. However it is not necessarily related to code and it is "social" in nature, i.e., connected to people and their development organisation. Paraphrasing Cunningham [6] who first introduced technical debt, social debt can be thought as: "not quite right development community - which we postpone making right".

While technical debt has received increased attention over the years, this other form of debt, namely "social debt" has remained latent and relatively unexplored. For instance, software engineering practitioners still lack a way to formalise socio-technical decisions and measure the debt (if any) connected to these decisions.

This paper discusses social debt pivoting around a provocative research question: *"what is social debt?"* First, we analyse scenarios from practice that can teach us what social debt is and what it is not. Second, we illustrate characteristics and challenges of social debt comparing with technical debt [5].

From the comparison with technical debt, we learned that social debt is, at first glance, erratic and unpredictable. A community could present an "ideal" structure and incur social debt later. Conversely, by discussing scenarios from practice we learned that a (precise) definition for social debt is far from being a simple matter of visualising a development community and studying its properties.

II. What Is Social Debt?

From sociological literature, quoting from Muir [7] "social debt of a society represents the set of strained social relationships that emerges as a consequence of debtor-creditor circumstances". In software engineering the same concept can be used, for example, to represent the lack of trust within a community [8] or the degree to which it is immature or unable to tackle a certain development problem. First, using four scenarios we observed in practice, we discuss possible characteristics of communities that suggest the presence of social debt. Finally, following the analogy with technical debt, we explore challenges connected to the study of social debt.

A. Is your community in debt?

In the following, we consider four simple scenarios we observed in software engineering industrial practice, to exemplify social debt. To instrument our discussion on the notion of social debt, we studied the concepts emerging from technical debt (see Fig. 1) and mapped our scenarios on the four quadrants, defining the symmetric concepts for social debt (see Fig. 2). Scenario 1 was tailored from [9]. Scenario 3 can be observed online, accessing the Apache Foundation community. Finally, we observed Scenario 2 and 4 while working with two of our industrial partners (their name is confidential).

Scenario 1 - A company uses communication filtering guidelines (e.g., knowledge-passing checklists to protect certain knowledge assets) to protect industrial secrets in the

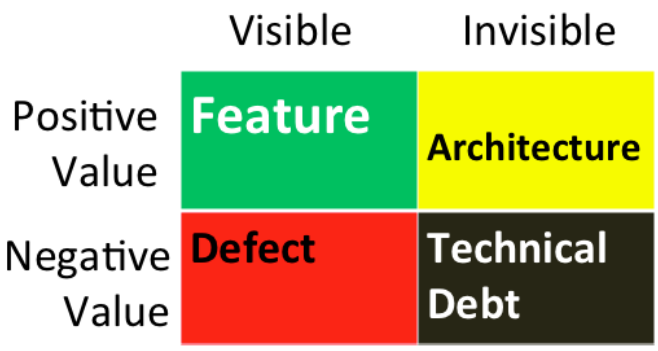

Fig. 1. Visualising Technical Debt: Invisible technicalities with negative value [5].

communication with outsourced partners. In this case the use of protection guidelines is an explicit socio-technical decision that produces a positive effect (protecting industrial secrets). The effect is a *feature* of the development community, something that is explicitly deployed and whose costs become a calculated risk. We do not observe any social debt in this case. The community *features* guidelines to protect industrial secrets. This effect is expected and its impact is constant, it doesn't introduce a cost which increases over time, rather saves money connected to loss of industrial secrets.

Scenario 2 - A company adopts Enterprise Social Networking (ESN) technologies to invest on IT-supported collaborative work. Using ESNs is a socio-technical decision, since it changes how people perceive their peers and changes the way tasks are allocated and carried-out. However, these socio-technical effects are both positive (increase awareness and reachability of people) and invisible (they are emergent properties of the social community of developers). These effects can be made evident by making explicit the *architecture* of the development community, i.e., the set of quantities (e.g., number of members, frequency of communication) and qualities (e.g., informality vs. formality, or situatedness vs. dispersion) that describe a community's structure and properties. The positive effect does not generate (or feed into) a social debt, rather it potentially increases the revenue stream from the developers' community behaviour by reducing collaboration and knowledge-retrieval times [10].

Scenario 3 - A company develops using three sites in collaboration with an open-source community. The informal, erratic frequency of open-source contributions is not consequent to any socio-technical decision. Rather the effect is an emergent *feature* of open-source communities. However, in collaboration with closed-source communities, this *feature* of open-source communities produces negative effects (e.g., unpredictable releases) within the mixed open-/closed-source community in our scenario. These consequences are visible to all within the development community. The negative effects however, are accepted and mediated through ad-hoc, explicit integration guidelines, e.g., [11]. The effect (originally a *feature* of pure open-source) becomes a *defect* when open- and closed-source are mixed. The explicit nature of the community *defect* does not create a social debt, on the contrary, the effect is visible and recognised and its negative side is mediated and worked upon through specific agreements. The development community adapts to the explicit presence of the *defect* using ad-hoc guidelines.

Scenario 4 - Finally, consider a scenario in which a company aims at renovating a legacy product where the original development knowledge and expertise went lost (e.g., people retired or architecture is eroded [12]). This is common practice in software engineering, where teams are dismantled "automatically" when the project is finished, often without explicit hand-overs or quality documentation. In this case, the socio-technical decision to renovate without the original community and its knowledge exposes an invisible and negative effect: a *social debt*. People, expertise and knowledge previously developing and sustaining the product are lost. The current community developing on the product is *sub-optimal*, and needs unplanned investment to equalise the debt (e.g., by reverse-engineering lost knowledge). The additional cost remains invisible until the company kicks off the modernisation project. Moreover, the cost increases linearly with the increased need for lost knowledge, i.e., as more parts of the system need to be modernised. A direct consequence of social debt is that company is likely to spend far more than planned.

B. How is your social debt different than your technical debt?

Technical debt studies ways to maintain the balance between rapid deployment and long-term value [5], [13]. Fig. 1 sketches the notion of technical debt. Works proposed under the technical debt banner provide practitioners with methods and metrics for its evaluation and tracking [14], [15].

Table I [1] contains an initial mapping between social debt and concepts/characteristics related to technical debt from [5], [6], [13]–[15]. Column 1 explores technical debt characteristics; Column 2 summarises technical debt challenges; Column 3 matches technical debt with related concepts we can observe for social debt while Column 4 summarises our observations and challenges on social debt. This exercise reveals that challenges for social debt on Table I can be clustered in three research areas: (a) measuring and predicting social debt; (b) visualising and analysing development community status; (c) establishing and mitigating the impact of social debt. All three areas can inherit much from technical debt both in methods and tools. Each of the areas presents many intriguing research questions, for example: (a) how can the development community be measured to find its social debt, if any?; (b) What aspects of a development community need to be made explicit to determine its performance?; and, (c) How can social debt be measured to calculate risks of evolvability and quality?

We recognise, however, that before exploring other research questions our main research question deserves a well defined and clear-cut answer: *What is social debt?*

[1] available online: http://www.fileden.com/files/2012/3/7/3275239/socdebt.pdf

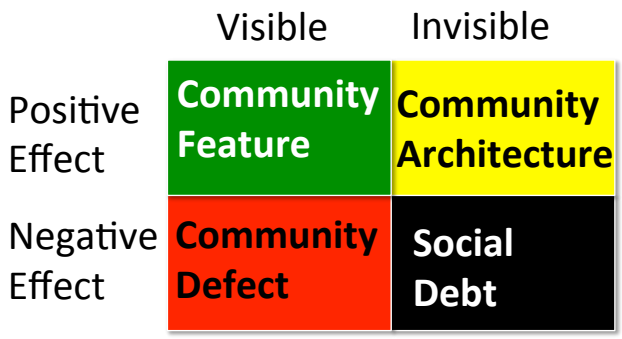

Fig. 2. Visualising Social Debt: invisible community properties with a negative effect for development.

III. Observations and Discussion

Following our previous discussion on the notion of social debt (see Section II), we made three key observations.

First, a definition for social debt is far from being a clear-cut financial figure connected to the configuration of development people. For example, the first and second scenario both show a visible effect that organisations produce or accept to proceed with development. However, communities are unpredictable and erratic, and there might be no simple way to establish and track the "stability" or "welfare" of this configuration. By using the framework in Fig. 2 we have intuitively categorised the decisions from said scenarios as a community *feature* and *bug* respectively but we have no way of knowing if the produced effect is indeed positive/negative and if it stays positive/negative. Consequently we can observe that social debt is much more dynamic and unpredictable than technical debt. Further discussion and study is needed to establish the boundaries of this dynamicity.

Second, while technical debt is residing within the system codebase and related artefacts, social debt seems more pervasive and intertwined with technical debt. For example, in scenario 4, a socio-technical decision (e.g., using additional people as consultants to help with product renewal) can generate both technical (e.g., missing or "hunch" based architectural reasoning from the consultants) and social debt (e.g., incremental uncooperativeness of development community with "technical experts" added to the game). Therefore, besides incurring technical debt, any socio-technical decision can produce within the community any or all the effects from Fig. 2. Consequently, we argue that the relation between technical and social debt deserves further study.

Third, mechanisms in technical debt endeavour to produce financial figures (i.e., value) to represent the debt. These financial figures are usually produced measuring code and other software artefacts. When reproducing Fig. 2 from Fig. 1, we deliberately used the word "effect" rather than "value" because we we need more research to understand what and how to measure or quantify the effects we exemplified in Section II. Additional investigation should focus on the causal relations behind the social effects we exemplified, before the word "value" can be used.

IV. Previous and Related Work

Nagappan et al. [16] show in practice the influence of organisational structure and other "human" aspects on software quality. This and similar works (e.g., [17] or [18]) bring evidence that motivates our study of social communities in organisations and the debt (if any) connected to them. This family of studies contributes to social debt by providing evidence of its existence and impact. In addition these studies provide valuable data to identify the orders of magnitude that regulated social debt.

Subsequently, studies on socio-technical congruence [19] can support the study of social debt. Socio-technical congruence is the degree to which technical and social dependencies match, when coordination is needed. Authors in [19] present socio-technical congruence in formal terms and empirically investigate its impact on product quality. Similar works (e.g., [20]), can be used as starting points to evaluate metrics for social debt. Perhaps socio-technical congruence leads the way in this path, representing a first rudimental metric for social debt in certain development communities. An evolution of the socio-technical stream leads to works such as [21] in which the authors discuss awareness maintenance mechanisms. These mechanisms are intuitively close to the notion of social debt, since their role is to track and maintain project knowledge with the aim of limiting delays and connected "debt". The study of industrial practice can start by applying in practice mechanisms such as those discussed in [21], analysing/comparing the community layout of different industrial cases.

Other works such as [22], use social-network analysis to investigate coordination among groups of developers with socio-technical dependencies. These works can inspire the usage of social-networks analysis tools and approaches to elaborate on social debt, studying the very foundations of communities, i.e., their social-network representation.

In addition, works in organizational decision-making (e.g., offshoring) can provide sample arenas in which social debt emerges. Understanding if the current organizational layout of a company is performant (or even compatible) with certain decision is vital to measure social debt for decisions [23].

Finally, many works similar to [23] (e.g., [24], [25] and [26], [27]) investigate the influence of organizational decisions on collaborations and product quality aspects, both in open- and closed-source ecosystems. These works can avail to study the impact of social debt in different scenarios, in terms of end-product quality and evolvability. Finally, works such as [28] provide ways to approach the measurement of social debt on many abstraction levels common in distributed IT risk management. These resources can steer and support the study around the notion of social debt.

V. Conclusion

This paper discusses the notion of social debt from multiple perspectives. We operated two analyses to avail our discussion of the notion: first, we investigated example scenarios from

practice that uncover some possible characteristics of social debt; second, we compared our understanding of social debt to technical debt, identifying possible research challenges.

We found that mapping social debt to technical debt leaves many interesting research questions open. These questions might lead to deepen software engineers' understanding of the many dimensions behind social debt in software engineering communities and how it can lead to failure. However, we learned that much work is still needed to properly define and study social debt in software engineering or its unforeseeable variations, e.g., over performing communities, working well also in presence of debt. In addition, we also learned that social debt is very dynamic in nature. This dynamic nature must be (semi-)formally defined and visualised before it can be studied and/or measured. Finally, we learned that "pinpointing" and isolating social debt is very difficult, given its interconnection with software people and artefacts.

The main contributions of this paper are twofold: first, we investigate scenarios from practice that lead to social debt "characteristics"; second we provide challenges within the study of social debt, by comparing it with technical debt.

In the future we plan to formalise social debt by observing real-life industrial software life-cycles. Moreover, we plan to devise mechanisms to visualise and study the social community structure, by putting together properties and observable characteristics from our previous work [2]–[4].

REFERENCES

[1] J. Keyes, *Social software engineering*. Boca Raton, FL: Taylor & Francis, Auerbach Series, 2011.

[2] D. A. Tamburri, P. Lago, and H. van Vliet, "Uncovering latent social communities in software development," *IEEE Software*, vol. 30, no. 1, pp. 29 –36, jan.-feb. 2013.

[3] D. A. Tamburri, E. di Nitto, P. Lago, and H. van Vliet, "On the nature of the GSE organizational social structure: an empirical study," *proceedings of the 7th IEEE International Conference on Global Software Engineering*, pp. 114–123, 2012.

[4] D. A. Tamburri, P. Lago, and H. van Vliet, "Organizational social structures for software engineering," *to appear in ACM Computing Surveys*, pp. 1–35, 2012.

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

[6] W. Cunningham, "The WyCash portfolio management system." *OOPS Messenger*, vol. 4, no. 2, pp. 29–30, 1993.

[7] D. E. Muir, "The social debt: An investigation of lower-class and middle class norms of social obligation," *American Sociological Review*, 1962.

[8] Moe, Nils Brede and ?mite, Darja, "Understanding a lack of trust in global software teams: a multiple-case study," *Software Process: Improvement and Practice*, vol. 13, no. 3, pp. 217–231, 2008.

[9] C. Manteli, B. van den Hooff, A. Tang, and H. van Vliet, "The impact of multi-site software governance on knowledge management," in *ICGSE*, 2011, pp. 40–49.

[10] M. Li, G. Chen, Z. Zhang, and Y. Fu, "A social collaboration platform for enterprise social networking." in *CSCWD*, L. Gao, W. Shen, J.-P. A. Barths, J. Luo, J. Yong, W. Li, and W. Li, Eds. IEEE, 2012, pp. 671–677.

[11] G. Walsh, M. Schaarschmidt, and H. F. O. von Kortzfleisch, "Harnessing free external resources: Evidence from the open source field." in *ICIS*. Association for Information Systems, 2012.

[12] B. Merkle, "Stop the software architecture erosion." in *SPLASH/OOPSLA Companion*, W. R. Cook, S. Clarke, and M. C. Rinard, Eds. ACM, 2010, pp. 295–297.

[13] P. Conroy, "Technical debt: Where are the shareholders' interests?" *IEEE Software*, vol. 29, no. 6, p. 88, 2012.

[14] J.-L. Letouzey and M. Ilkiewicz, "Managing technical debt with the sqale method." *IEEE Software*, vol. 29, no. 6, pp. 44–51, 2012.

[15] R. Marinescu, "Assessing technical debt by identifying design flaws in software systems." *IBM Journal of Research and Development*, vol. 56, no. 5, p. 9, 2012.

[16] N. Nagappan, B. Murphy, and V. Basili, "The influence of organizational structure on software quality: an empirical case study," in *International conference on Software engineering*. Leipzig, Germany.: IEEE, May 2008, pp. 521–530.

[17] A. Repenning, N. Ahmadi, N. Repenning, A. Ioannidou, D. Webb, and K. Marshall, "Collective programming: making end-user programming (more) social," pp. 325–330.

[18] D. Viana, T. Conte, D. Vilela, C. R. B. de Souza, G. Santos, and R. Prikladnicki, "The influence of human aspects on software process improvement: Qualitative research findings and comparison to previous studies," in *EASE*, 2012, pp. 121–125.

[19] M. Cataldo, A. Mockus, J. A. Roberts, and J. D. Herbsleb, "Software dependencies, work dependencies, and their impact on failures." *IEEE Trans. Software Eng.*, vol. 35, no. 6, pp. 864–878, 2009.

[20] I. Kwan, A. Schroter, and D. Damian, "Does socio-technical congruence have an effect on software build success? a study of coordination in a software project," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 307–324, May 2011.

[21] C. R. B. de Souza and D. F. Redmiles, "The Awareness Network, To Whom Should I Display My Actions? And, Whose Actions Should I Monitor?" *IEEE Trans. Software Eng.*, vol. 37, no. 3, pp. 325–340, 2011.

[22] C. Bird, N. Nagappan, H. Gall, B. Murphy, and P. Devanbu, "Putting it all together: Using socio-technical networks to predict failures," in *Proceedings of the 2009 20th International Symposium on Software Reliability Engineering*, ser. ISSRE '09. IEEE Computer Society, 2009, pp. 109–119.

[23] J. J. Cusick and A. Prasad, "A practical management and engineering approach to offshore collaboration." *IEEE Software*, vol. 23, no. 5, pp. 20–29, 2006.

[24] C. Bird, N. Nagappan, P. Devanbu, H. Gall, and B. Murphy, "Does distributed development affect software quality? an empirical case study of windows vista," in *Proceedings of the 31st International Conference on Software Engineering*, ser. ICSE '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 518–528.

[25] M. Pinzger, N. Nagappan, and B. Murphy, "Can developer-module networks predict failures?" in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, ser. SIGSOFT '08/FSE-16. New York, NY, USA: ACM, 2008, pp. 2–12.

[26] A. Meneely and L. A. Williams, "Secure open source collaboration: an empirical study of Linus' law." in *ACM Conference on Computer and Communications Security*, E. Al-Shaer, S. Jha, and A. D. Keromytis, Eds. ACM, 2009, pp. 453–462.

[27] B. Witten, C. Landwehr, and M. Caloyannides, "Does open source improve system security?" *IEEE Software*, vol. 18, no. 5, pp. 57–61, Sep. 2001.

[28] R. Prikladnicki, J. R. Evaristo, J. L. N. Audy, and M. H. Yamaguti, "Risk management in distributed it projects: Integrating strategic, tactical, and operational levels," *IGI Global: International Journal of e-Collaboration*, vol. 2, no. 4, pp. 1–18, 2006.