When Software Architecting Leads to Social Debt

Damian A. Tamburri
VU University Amsterdam
Amsterdam, The Netherlands
Email: d.a.tamburri@vu.nl

Elisabetta Di Nitto
Politecnico di Milano
Milan, Italy
Email: elisabetta.dinitto@polimi.it

***Abstract*—Social and technical debt both represent the state of software development organizations as a result of accumulated decisions. In the case of social debt, decisions (and connected debt) weigh on people and their socio-technical interactions/characteristics. Digging deeper into social debt with an industrial case-study, we found that software architecture, the prince of development artefacts, plays a major role in causing social debt. This paper discusses a key circumstance wherefore social debt is connected to software architectures and what can be done and measured in response, as observed in our case-study. Also, we introduce DAHLIA, that is "Debt-Aimed arcHitecture-Level Incommunicability Analysis" - a framework to elicit some of the causes behind social debt for further analysis.**

I. Introduction

In layman's terms, social debt is the additional project cost in the current state of affairs caused by sub-optimal socio-technical decisions [1]. The additional cost may be connected to a wide variety of possible decisions, e.g., changing the organisational structure [2] of the development network [3] (e.g., through outsourcing), changing the development process (e.g., by adopting agile methods), leveraging on (too much or too little) global collaboration across networked development organisations. Although an established body of work discusses technical debt and its relations with software architecture and its qualities, social debt remains relatively unexplored.

In an effort to explore and better define social debt, we conducted an explorative industrial case-study.

We found that software architecture, the prince of software development artefacts, along with the very process of architecting, play a key role in generating but also pinpointing some of the accumulated social debt. More in particular, *architecture decisions can take place "by osmosis"*, that is, considering information that crosses every possible communication link across the development and operations network. In this circumstance, loss of essential information is almost inevitable. For example, architecture decisions in this circumstance can emerge from the following sequence of events: (a) clients encounter difficulties; (b) these difficulties are communicated to operators; (c) operators discuss and alert developers; (d) developers force architects into an architectural change, sometimes making partial architecture decisions of their own. The problem is that much information, rationale and requirements can go lost in the resulting communication chain. We observed many nasty circumstances linked to this chain of events, such as architecture erosion [4], poor architecture documentation, lack of vision and, eventually, mistrust.

To tackle the above circumstance, we introduce DAHLIA, that stands for "Debt-Aimed arcHitecture-Level Incommunicability Analysis", a metrics framework to evaluate the communicability of decisions across a development network [3]. DAHLIA uses basic social-network analysis (SNA) assumptions and techniques [5] to elicit the social debt connected to (some of) the patterns above. We illustrate DAHLIA using architecture decisions from our case-study.

We conclude that studying social debt at the architecture level is as important as studying its technical counterpart to reduce waste during the software lifecycle. Also, social and technical debt are deeply intertwined in both cause and effect. Software architectures are viable ways to study this relation, unravel its implications on software products and use it to improve lifecycle and product quality.

II. Architecting By Osmosis Leads to Social Debt

The results in this article are based on a study in a large IT service provider (which we call *Capita* from now on) for the aviation industry. *Capita* has around 3,000 employees in several locations in Germany and around Europe. Also, *Capita* controls several offices in 14 other countries.

The data we used to obtain our results is based on 16 semi-structured interviews[1] (with an average of 90 mins per interview). The study involves a total of 13 people some of which were interviewed multiple times, including managers, architects, developers, operators and Integration engineers. Interviews[2] were structured according to procedures and guidelines in [6]. It should be noted that social debt itself was never mentioned during data elicitation, to avoid bias. Following strict non-disclosure agreements, all transcriptions were completely anonymised at the source. Our material was analysed through Grounded-Theory [7].

As previously mentioned, we found recurrent series of circumstances in which architecture decisions and the process of architecting reportedly generated social debt. Essentially these circumstances represent architecting patterns [8], to be avoided if social debt is not acceptable. In our case study, we saw recurring a peculiar series of events leading to what we call "architecting by osmosis".

In layman's terms, osmosis refers to the process of permeating a solvent through a semi-permeable (series of) membrane(s)[3]. By comparison, architecting by osmosis means

[1] a summary of key interviews is available online: http://tinyurl.com/ljqgay9

[2] interview guide is protected by non disclosure agreements but can be made available upon written and signed request.

[3] "Osmosis". Oxford English Dictionary (3rd ed.). Oxford University Press. September 2005.

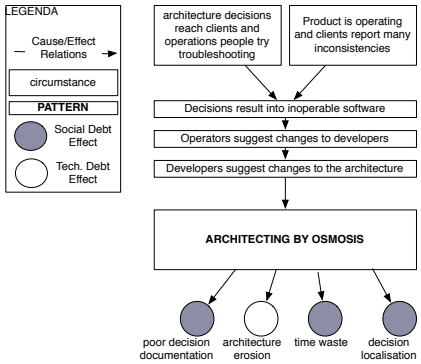

Fig. 1. Architecting by osmosis, semi-permeable communication.

making architecture decisions using knowledge that is filtered through many semi-permeable communication links. We observed architecting by osmosis manifesting when the following sequence of events occurs: (1) the effects of certain decisions reach clients and product operators but result in inoperable software; (2) operators, pushed by clients, share malcontent with developers and suggest technical changes; (3) developers evaluate (and sometimes partially implement) possible technical changes and suggest change to architecture decisions; (4) architects make necessary changes in decisions with knowledge that was partially filtered by all communication layers in the development network. This pattern is depicted in Fig. 1. It was found recurring over time and was encountered 3 times. Some of the most common consequences we found resulting from this pattern are: (a) decision localisation - architecture changes were made by certain architects who did their best to communicate but the decision (and consequent refactoring) remained local to their own team - apparently architects assumed that the same communication chain for the change request would disseminate the change itself - eventually this phenomenon resulted in uncooperative behaviour; (b) poor decision documentation - much of the rationale and reasoning behind architecture change requests went lost as people filtered out details through every communication link, this reportedly caused frustration when architecture documents were involved and eventually led to deprecation of architecture documents; (c) architecture erosion - some architecture decisions were changed as they created unstable configurations, this reportedly caused a great deal of misalignment.

III. Paying Back Social Debt?

A large part of the negative and invisible effects reported in Section II were mitigated with success in *Capita* using a practice referred by people part of *Integra* as an *Architecture Board* (AB). An AB is essentially a sub-community in *Integra* comprised of people who were, are or are likely to be responsible for architecture decisions and their dissemination. More in particular, any person involved in the project and matching the following prerequisites would become AB member:

- The person has made or influenced architecture decisions with provable rationale;
- The person has architecture expertise or concerns and belongs to a site without an AB member;
- The person has architecture expertise or concerns and belongs to a team without an AB member;

Also, for certain architecture decisions requiring particular domain-specific expertise, client or domain-specific analysts' intervention was required for one or more AB meetings.

Thus structured, an architecture board is consistent with the Problem-Solving Community (PSC) we previously reported in [2]. The community met on a bi-weekly basis to fulfil two organisational goals: (a) make and disseminate architecture decisions; (b) gather and monitor the usage of organisational culture inherent to devising and disseminating decisions.

This practice reportedly reduced debt connected to 3 of the patterns reported above, namely: (a) obfuscated architecting; (b) invisible architecting and (c) lonesome architecting.

Nevertheless, using ABs was not without nasty consequences. For example, some board-members eventually assumed subversive behaviour of their own, e.g., pulling decisions and decision-making towards their own concerns while disregarding or belittling others, with consequent emergence of social debt. Quoting from our interviews "[members of the board] are essentially different architects from different teams and pulling towards their own direction instead of finding a common standard [for organisational structure and goals]". We found reports of this circumstance 4 times. Also, the AB itself received little or no formal recognition by the organisational structure around *Capita*. This reportedly compromised its existence. We saw indication of this 5 times in our interviews. For example, one architect reported that " The architects board is also unofficial.. [this means] no formal organization, no discipline. [Hence] still there are cases in which architects are not involved and requirements are not well documented and specified. [...] Involvement of architects always! To minimise the risk of false implementation".

IV. Social Debt and Architecting: What can be measured?

Analysing our interviews, the resulting patterns and their root-cause we observed that sharing architecture knowledge, i.e., the act of making available software architecture decisions and related artefacts [9], is necessary for the quality of software processes and resulting products but it is not sufficient. It is in fact *architecture incommunicability* that plays a key role in the emergence of social debt. ***Architecture incommunicability is the inability to communicate architecture decisions tempestively to those who should be aware of them due to adverse organisational or social circumstances across the development network***. In other words, *architecture incommunicability* is the likelihood that who should know about architecture decisions actually does not know anything about them. By its very nature, *incommunicability* has to do with communication and hence is afflicted by social and organisational circumstances (e.g., organisational filtering protocols or non-disclosure agreements). As such, however, *architecture incommunicability* can be studied combining principles and techniques from social-networks analysis (SNA) with an analysis of who makes architecture decisions, as opposed to who actually knows about said decisions. Our scenario seems consistent with what is known as *"weak-ties hypothesis"* as elaborated by Granovetter in [10] in

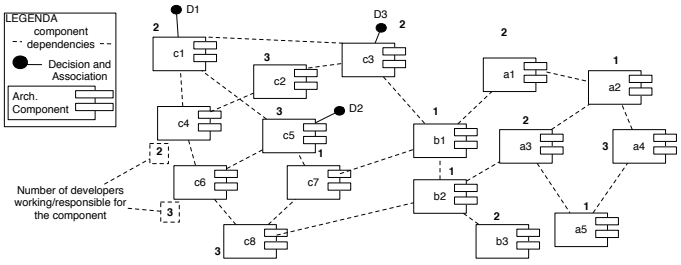

Fig. 2. Structural dependency view of software architecture in *Integra*.

SNA. Paraphrasing from [10], the weak-ties hypothesis implies that:

> *"if entity A is linked to both B and C then there is a considerable probability that B and C are related as well, with pretty much the same relation".*

More in particular, we noticed that mainly people who are *strongly-tied* with the decision maker are actually made aware of the decision in a short time-span. Conversely, people with *weak-ties* either with decision-makers or their strongly-related partners find out about architecture decisions only eventually, after a longer time-span.

Under this hypothesis, analysts can observe and compute the difference between: (a) the number of people who are among the decision-maker's strong-ties; (b) the number of people who should actually know certain decisions. This difference can be seen as a rough estimate of *architecture incommunicability*. We can also hypothesise that AB's were effective in *Integra* since they enabled weak-ties in the board to be used as architecture knowledge conveyors.

A. DAHLIA: Measuring Architecture Incommunicability

Stemming from the above concepts and definitions of *architecture incommunicability* we defined DAHLIA, that is "Debt-Aimed arcHitecture-Level Incommunicability Analysis".

DAHLIA was defined using GQM [11] through the following logic:
GOAL: measure the likelihood that everyone knows about decisions of which they should be aware and evaluate how much does the delay connected to unawareness cost;
QUESTION: *"who else should know about a certain decision D besides the decision-maker? who actually knows about D after the decision has been made and preliminarily disseminated?"*
METRIC 1: given a certain decision D, understand which components are related to it and infer who is currently working or responsible for said components. This measurement is a percentage of people in the development network, i.e., a *decision popularity metric (DEM)*.
METRIC 2: given a certain decision D, understand the subgraph of the development network that is aware about a certain decision assuming the strong ties hypothesis [10] for developers related to the decision-maker. This is again a percentage of people in the development network , i.e., a *decision awareness metric (DAM)*.
METRIC 3: given a DEM and its related DAM, compute the gap between the two - this amount roughly indicates the percentage of people across the development network that should know but are almost certainly unaware of decision D, i.e., the *mean architecture incommunicability (MAI)* indicates the mean percentage of people that should be aware but, in fact, are not.
METRIC 4: compute the *Average Per-PersoN Delay (APPeND)* connected to decision unawareness, this is a measure of time, and can be measured in man-hours. Once the number is available, multiply APPeND (mean cost) and MAI (mean number of unaware developers) values to elaborate a rough estimate of social debt.

The above reasoning is conceptually similar to the logic behind *Transactive-Memory Systems* (TMS) previously defined in social-networks and organisations research by Wegner et Al. [12]. Essentially, TMS are mental mechanisms through which groups collectively encode (by capturing actors and actions, "who did what"), store (by capturing the experts, "who should know"), and retrieve knowledge (by retrieving the experts, "who knows what") [13], contributing to the creation of a group-mind, or *collective intelligence* [13].

In layman's terms, DAHLIA is a framework to measure communicability using a rudimentary TMS of architecture decisions. This rudimentary TMS can be obtained combining two socio-technical artefacts related to software architecture and computable during software lifecycles as follows:

1) A *structural dependency view* of the software architecture components augmented with: (a) a mapping of architecture decisions onto related components; (b) a mapping of how many developers/operators are currently working or responsible for said components. This view relates architecture components among themselves (using their associations or dependencies) with decisions concerning said components. A sample of this view computed using data from our case-study can be seen in Fig. 2. *This view is needed to measure how many people should know about a certain architecture decision (and the connected DEM), as defined above*.
2) A *social-network representation of the development network* computed using strong interaction and collaboration relations existing among developers.

Applying DAHLIA means following this simple process:

1) Evaluate and subtract DEM and DAM for architecture decisions taken up to the instant under investigation;
2) Compute an average of the above subtractions, i.e., Mean Architecture Incommunicability (MAI);
3) Compute the APPeND value for the project at hand;
4) Multiply APPeND and the number of people reflected by MAI, i.e., additional project cost connected to social debt;

Note that the above process can be applied even with an a-posteriori evaluation. In this case observers evaluate and subtract DEM and DAM for every architecture decision made during the observed project.

To illustrate DAHLIA, Section IV-B applies the framework in practice, using data and models[4] from our case-study.

B. DAHLIA in Action: a Scenario from our Case-Study

Using data from our scenario we were able to apply DHALIA in practice for a limited number of decisions (three decisions) both before and after the usage of Architecture Boards. This allows us to: (a) illustrate DHALIA and its workings on a very limited and simple yet expressive scenario; (b) illustrate a rough estimate of the effect that using Architecture Boards introduced in our scenario.

First, let us assume D1 is a decision associated to component C1. Also, let us assume that component C1 is related to components C3, C4 and C5 respectively. *What is the incommunicability for D1?*

According to our definitions from Section IV-A and the assumptions above, there are at least 8 people who should be made aware of D1, namely:

- 1 developer who is currently working on the same architecture component on which the decision is affecting;
- 7 developers working on components related to the one affected by D1;

8 out of 13 people means that 61% of the development network should be aware of D1. This is our DEM metric. However, by virtue of the strong-ties hypothesis [10], people in strong-ties with the decision-maker are more likely to be made aware about D1. According to our data, in our scenario there are 5 such people strongly tied to "Arch. 3", hence, about 38% of the development network (this percentage is our DAM) is more likely to know of the decision in a shorter time.

By virtue of the weak-ties hypothesis, subtracting the two numbers gives a very rough estimate of how many people should know but actually do not in a short time span, i.e., the MAI value sought for: 8 - 5 = 3, i.e., around 23% of the network. These people are more likely to know of D1 eventually after a longer time span.

We iterated this exercise on our data for *Integra* for available decisions and found that, on average, 2 people part of the project were not aware of architecture decisions as they were taken or changed. Now, let's assume that the patterns causing decision unawareness yield an average delay of 4 man-hours (our APPenD value) as we observed in [1]. What results is 1 Man-Day additional cost, connected to a MAI value of 23%. This delay is connected to social debt across *Integra*.

V. Conclusion

This paper reports on patterns in the process of architecting found to result in what we call social debt, i.e., a picture of the current state of things burdened by sub-optimal socio-technical decisions. This paper also reports on a practice observed in our case-study to mitigate some of the nasty consequences connected to emerging social debt. Finally, we outlined DAHLIA, a sample metric for architectures to make (some) social debt explicit by measuring architectural (in)communicability, i.e, the likelihood that the developers' network is (un)aware of architecture decisions.

We observed that social debt and software architectures are tightly knit together and with technical debt as well, and deserve further attention in the future. We plan to elaborate on our findings, expanding our understanding of the relations between social debt, its technical counterpart and software architectures possibly with an immersive study in industry. Also, we plan to understand the differences between social debt in closed-source industries (as reported in this paper) and open-source, to possibly find successful architecting patterns elicited from open-source, if any. In addition, we plan to elaborate further on the proposed metric for software architecture communicability, demonstrating its representation condition and applying it in practice to provide more solid validation. Finally, we are already setting up an industrial study to compare different architecting practices in the scope of social debt, e.g., to evaluate the efficacy of said practices also in reducing debt-generating decisions[5].

[4]All data and models were properly anonymised, reshuffled and modified according to our Non-Disclosure Agreement.

[5]This research has been partially supported by the European Commission, Grant no. FP7-ICT-2011-8-318484 MODAClouds project.

References

[1] D. A. Tamburri, P. Kruchten, P. Lago, and H. van Vliet, "Social debt in software engineering: Insights from industry," *Journal of Internet Services and Applications*, pp. 1–17, Under Review 2014.

[2] D. A. Tamburri, P. Lago, and H. van Vliet, "Organizational social structures for software engineering." *ACM Comput. Surv.*, vol. 46, no. 1, p. 3, 2013.

[3] A. Meneely and L. Williams, "Socio-technical developer networks: should we trust our measurements?" in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE '11. New York, NY, USA: ACM, 2011, pp. 281–290.

[4] D. E. Perry and A. L. Wolf, "Foundations for the study of software architecture," *SIGSOFT Softw. Eng. Notes*, vol. 17, no. 4, pp. 40–52, Oct. 1992.

[5] M. Kilduff and W. Tsai, *Social Networks and Organizations*. Sage Publications Ltd, 2003.

[6] G. Neville-Neil, "Interviewing techniques." *ACM Queue*, vol. 9, no. 6, p. 30, 2011.

[7] J. Corbin and A. Strauss, "Grounded theory research: Procedures, canons, and evaluative criteria," *Qualitative Sociology*, vol. 13, no. 1, pp. 3–21, 1990.

[8] W. J. Brown, R. C. Malveau, H. W. S. McCormick, and T. J. Mowbray, *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis: Refactoring Software, Architecture and Projects in Crisis*, 1st ed. John Wiley & Sons, 1998.

[9] J. F. Hoorn, R. Farenhorst, P. Lago, and H. van Vliet, "The lonesome architect." *Journal of Systems and Software*, vol. 84, no. 9, pp. 1424–1435, 2011.

[10] M. Granovetter, "The strength of weak ties," *American Journal of Sociology*, no. 78, pp. 1360–1380, 1973.

[11] V. R. Basili, G. Caldiera, and D. H. Rombach, *The Goal Question Metric Approach*. John Wiley & Sons, 1994, vol. I.

[12] D. M. Wegner, "A computer network model of human transactive memory," *Social Cognition*, vol. 13, pp. 1–21, 1995.

[13] D. M. Wegner, T. Giuliano, and P. Hertel, *Cognitive interdependence in close relationships*. New York: Springer-Verlag, 1985, pp. 253–276.

[14] R. L. Nord, I. Ozkaya, P. Kruchten, and M. Gonzalez-Rojas, "In search of a metric for managing architectural technical debt." in *WICSA/ECSA*. IEEE, 2012, pp. 91–100.

[15] 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.