TechDebt @ ICSE 2025, Extended Abstract

Experiences with Technical Debt and its Root Causes in a Merged Industrial Company

Magnus Drewsen Jørgensen
Mjølner Informatics A/S
Aarhus, Denmark
mdj@mjolner.dk

Nina Wiborg Mølgaard
Mjølner Informatics A/S
Aarhus, Denmark
nwm@mjolner.dk

Morten Jokumsen
Mjølner Informatics A/S
Aarhus, Denmark
mjo@mjolner.dk

Jens Bæk Jørgensen
Mjølner Informatics A/S
Aarhus, Denmark
jbj@mjolner.dk

Henrik Bærbak Christensen
Aarhus University
Aarhus, Denmark
hbc@cs.au.dk

***Abstract*—Technical Debt (TD) is crucial to address in industrial software development. We have described and analysed TD within an industrial company that is the result of mergers and acquisitions of several other companies. Our results are (1) first-hand reports of TD directly from developers and software architects, (2) a categorization of various types of TD in architectural anti-patterns, and (3) an identification and discussion of possible root causes of the TD.**

Keywords—technical debt, organization, human factors, architecture, communication, cooperation

Introduction

We have examined Technical Debt (TD) at a large industrial company which has thousands of employees and serves in the magnitude of one million customers. The company sells subscriptions within the utility sector, like heating, gas, oil, electricity or similar. The company is only a few years old, and it is the result of mergers and acquisitions of many companies. To safeguard confidentiality, we keep the company as well as the type of utility supplied anonymous. This company will be referred to as UtilComp.

We report on experiences from a two-year period, from summer 2022 to summer 2024. Within this timeframe, approximately 80-100 people have worked on the software development under consideration, with an estimated total effort of 250,000-300,000 person hours. We have participated in developer and architect roles, and we have worked on two different but related projects:

(1) The Customer Entity Management project: A project made to address some of UtilComp's challenges related to handling scattered Customer Entity records from multiple external source systems of the merged companies. The main goal was to make single access point integrations from 'Source data holding systems' to self-service- and sales-platforms within UtilComp.

(2) The Better Digital Channels project: A project made to enhance the digital channels facilitating online interaction between UtilComp and its customers. The main goal was to deliver: (i) a new and improved web page, where UtilComp could sell subscriptions, (ii) a new and improved self-service web solution where UtilComp's existing customers could get an overview of their engagement, pay bills etc., and (iii) a brand-new app providing various overviews to customers.

The Better Digital Channels project was highly dependent on deliveries from the Customer Entity Management project, so communication between the two projects was necessary. Despite their dependency, the two project were in different parts of the organization, see Figure 1.

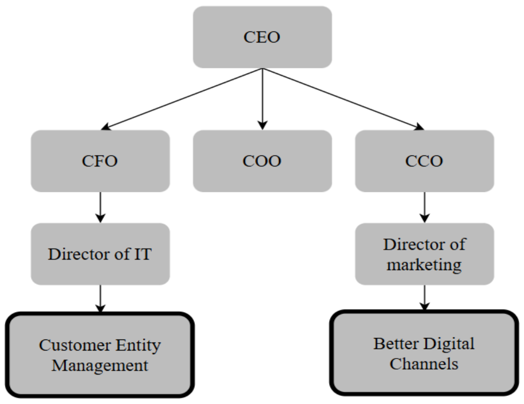

Figure 1: Organizational diagram of UtilComp's upper-level management layers

Method

We report our experiences. We have not done a case study or controlled experiment with a hypothesis and systematic data gathering to validate or invalidate it. We have carried out work, and we present a retrospective analysis of what was done and what was observed. Further, we rely on the results

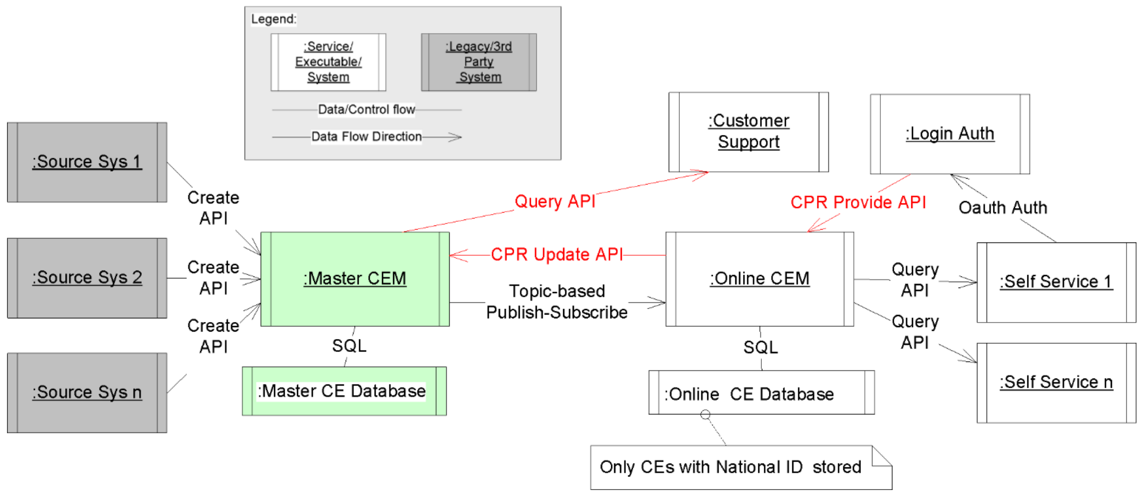

Figure 2: Customer Entity Management (CEM) architecture relevant for our analysis

from an ATAM analysis of UtilComp's software architecture that we have previously carried out.

RESULTS

We have identified examples of different types of TD that were found in both projects, and classified these in architectural anti-patterns.

Figure 2 shows an example of a relevant architecture extract for the Customer Entity Management project. The figure is there to give an overall impression; the details in the figure are not further described, due to limited space.

Major anti-patterns that we have seen in both projects are: (1) *Cyclic dependencies*; (2) *too many technologies*; (3) *high coupling to external systems*.

Examining how anti-patterns arose within the projects' system architectures, and examining UtilComp's organization and software setup, we have made qualified conjectures of root causes within the UtilComp setup causing these anti-patterns. The major root causes conjectured are:

(1) *Organizational distance* between software projects within UtilComp, causing insufficient or no communication channels between interrelated projects.

(2) *Unclear roles and responsibilities* between system architects, leading to unclear responsibilities of systems and duplicated functionality across systems.

(3) *Missing or unclear diagramming and documentation,* causing misalignment between systems.

(4) *Undesired side-effects of Scrum*, creating an emphasis on single team-tasks and short-term planning, rather than communication between teams and long-term planning.

(5) *Low level of digital maturity* in the quite young company UtilComp, where the significance of software development and avoiding anti-patterns like those mentioned above were not well recognized (and where we as software professionals did not argue the case well enough).

(6) *Being a result of many mergers* of companies, UtilComp must consolidate software and data from each of the merged companies, affecting the overall architecture.

In Figure 3, we have sketched conjectures about possible relations between the *Cyclic dependencies anti-pattern* and its possible root causes.

CONCLUSION AND OUTLOOK

We have first-hand observed and described examples of TD from a major industrial project and classified TD types in anti-patterns, and considered root causes. Our work would benefit from a careful analysis of its relation to the already established body of knowledge about TD, as exemplified by the reference list.

REFERENCES

[1] T. Sharma and D. Spinellis, "A Survey on Software Smells," *Journal of Systems and Software*, vol. 138, pp. 158-173, 2018.

[2] P. Avgeriou, P. Kruchten, I. Ozkaya and C. Seaman, "Managing Technical Debt in Software Engineering," in *Technical report, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, Rep. Dagstuhl Seminar 16162, 2016*, 2016.

[3] R. Verdecchia, P. Kruchten and P. Lago, "Architectural Technical Debt: A Grounded Theory," in *Software Architecture: 14th European Conference, ECSA 2020, L'Aquila, Italy, September 14–18, 2020, Proceedings*, 2020.

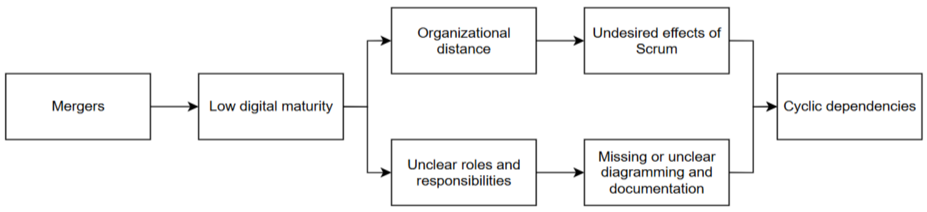
Mergers
Low digital maturity
Organizational distance
Undesired effects of Scrum
Unclear roles and responsibilities
Missing or unclear diagramming and documentation
Cyclic dependencies

Figure 3: Cyclic dependencies – cause and effect