2019 IEEE/ACM International Conference on Technical Debt (TechDebt)

Identifying Scalability Debt in Open Systems

Geir Kjetil Hanssen
SINTEF Digital
Trondheim, Norway
ghanssen@sintef.no

Gunnar Brataas
SINTEF Digital
Trondheim, Norway
gubra@sintef.no

Antonio Martini
Department of Informatics
University of Oslo, Norway
antonima@ifi.uio.no

***Abstract*—Architectural technical debt can be generated by changes in the business and the environment of an organization. In this paper, we emphasize the change in scalability requirements due to new regulations. Scalability is the ability of a system to handle an increased workload. For complex systems that are abruptly exposed via open interfaces and hence a greater workload, the scalability requirements may quickly increase, leading to technical debt. We term this scalability debt. This paper describes scalability triage, a light-weight, novel technique for identifying scalability threats as a form of technical debt. We illustrate this technique with an open banking case from a large software organization. Open banking is partly caused by the new European PSD2 regulative that enforce banks to open interfaces to unknown third-party actors. Banking systems are well-established, mature systems. However, with the advent of open banking and PSD2, the workload may quickly rocket. This leads to tougher scalability requirements and accumulated architectural debt, despite previously sound architectural decisions. Using scalability triage, such risks may be identified fast. It will then be possible to prevent this form of technical debt with timely reengineering.**

***Keywords*—agile development, scalability requirements, scalability triage, open banking**

I. Introduction

Architectural technical debt (ATD) can be avoided by an upfront design or careful maintenance of the system. However, recent research suggests that part of the ATD might be unknown and impossible to predict at the time of major architectural decisions [1]. For example, such ATD might be accrued by the evolution of requirements and other business or technological factors that are external to companies. However, although a few studies on ATD have been conducted [2], it is still challenging to avoid or timely tackle ATD that is not accumulated by a strategic decision, but it is accrued as a result of an external event that is outside of control for the organization.

In this paper we consider ATD originating from stricter scalability requirements following an abrupt opening of interfaces to a system that has previously been under control. Scalability is defined as the ability of a system to increase its capacity by consuming more resources. When these (legacy) systems were designed, scalability was sufficient, but because of an increased expected workload, scalability requirements became tougher and this has led to ATD. We term the presence of ATD related to scalability as scalability debt, a sub-category of ATD, and define scalability debt as: "Architectural technical debt originating by tougher scalability requirements because of substantial changes in the business (environment)".

Our goals in this paper are: *(1) To advocate for the necessity to study novel and lightweight approaches to manage scalability debt in open systems. And, (2)* To provide a first preliminary study of a lightweight analysis method that can support managing scalability debt in open systems.

The subsequent RQ is: *How can we identify, estimate and prioritize scalability debt in open systems?*

We have followed a case where a large banking software provider, named BankTech in this paper, now are facing scalability threats caused by new European regulations – the PSD2 directive which enforce all banks and related financial institutions to offer APIs to enable third-parties to build new FinTech solutions. This represents an abrupt change in the systems external environment. The goal of this new regulation is to strengthen the open European market and to enable new and better solutions for the end users. In this case, it may lead to a much higher pressure on existing banking systems in cases of many new third-party providers.

Through our case study we have found that such disruptive changes in the external environment of mature legacy systems, such as banking systems, may reveal technical debt in the form of scalability requirements, for systems, which previously have been well-known and under control. This control has been based on a very good overview of who the users are and how- and how-much they are using the system – this situation is now becoming highly uncertain and imposes a risk of failing scalability, which again would be bad for business, and eventually may lead to substantial fines from regulatory bank organizations.

Our underlying assumption is that scalability analysis both during design as well as scalability testing is time-consuming and costly. Especially in a setting where time-to-market is an important competitive factor, speed is essential. We therefore need a simple and fast technique where we capture scalability risks early. This gives focus on the scalability analysis task and as a result, this overwhelming task becomes manageable.

We share insights on a light-weight scalability-risk analysis technique that was developed to identify and estimate the impact of potential scalability debt fast and to prioritize necessary changes to the system as early as possible. We provide insights into the technique and early results that will guide further investigations.

II. Scalability

Capacity is the maximum workload a system can handle while still fulfilling its quality thresholds. *Scalability* is defined as the ability of a system to increase its capacity by consuming more resources [3-5]. Capacity and scalability again build on the following categories of scalability requirements:

- Planning horizon. Decide how long into the future you want to explore the scalability of your system, e.g., 2 or 5 years from now on?

- System boundaries. Specify the services and corresponding resources within the system boundaries, which should be included when measuring the quality of the system.
- Operations. Identify unique ways of interacting with the system, e.g., functions, calls, transactions, queries, and jobs, such as a payments or balance transaction.
- Quality metrics. Define precisely how to measure a certain performance quality. A typical metric in our domain is the 90th percentile of response times.
- Quality thresholds. Describe the boundary between acceptable and non-acceptable quality for each operation, using a quality metric, e.g., 1 second for the 90th percentile of response times.
- Work. Characterize what is done each time we invoke an operation and eventually determine resource demands for this operation. Work is related to the amount of data to be processed, stored, or communicated. We are interested in the highest values for the work parameters, which typically occur towards the end of the planning horizon.
- Burstiness. Estimate the fraction of peak load to average load during a given period. For example, a burstiness of 3 for the hours of a day, means that in the busy hour, the load is three times the average load during a day.
- Load. Describe how often an operation is invoked. For an open system, load is determined by an arrival rate, e.g., transactions per second. In a closed system, load is captured in terms of the number of end-users and their average think time. We are interested in the highest load values, i.e., load in the busiest hour, in the busiest day, in the busiest week, and in the busiest year in our planning horizon.
- Workload. The product of work and load.
- Critical operations. Operations where the product of work and load has a risk of not fulfilling the quality thresholds.
- Resources. Specify cloud and/or hardware resources: CPUs, disks, and networks. Software license costs are also part of the resources.
- Data consistency. Describe how up-to-date different replicas should be. Since replication is the key pattern for achieving scalability, strict consistency requirements will make it much harder to achieve a given level of scalability.

Unless a system is embarrassingly parallelizable, it will generally be too costly to make a completely scalable system. In other words, gold plating is not a good strategy. When demand for scalability increases and the system is not designed to support such demands, the system automatically accrues scalability debt. Scalability requirements have been mentioned in [6] and [12] when evaluating ATD, although without referring to specific sub-requirements.

III. Open Banking and PSD2

Both open banking and PSD2 (Payment Service Directive 2) are related to "fintech". The focus of PSD2 is for the bank to open up their services to third party providers (TPP). All providers which are authorized by a national financial body in Europe can become a TPP. PSD2 requires some open services from all banks, for example, payment and balance. Open banking is independent of PSD2. Open banking may go beyond this to also open up other services. PSD2 will be in effect from March 2019, starting with a six-months trial period.

It is important to know that TPPs need approval from customers, not from banks. Many customers have accounts in several banks. TPPs may offer a great variety of financial services, for example, financial guides explaining what money has been spent on. By getting approval from customers, these TPPs will also get access to customer data.

As part of PSD2, the quality for the TPPs using an open interface should be the same as for own customers using a closed interface. This quality requirement applies both to the response times and to freshness of the data (consistency). This means that even with high load because of many customers accepting TPPs, banks still have to provide acceptable response times. When a new TPP, for example, a part of Facebook, becomes popular, this may happen quite fast, in contrast to the quite modest and controllable load increase which, until now, has been the norm in financial services. This may therefore lead to abrupt increases in scalability debt.

IV. Case and Context

BankTech is a PSD2 provider for customers in the Nordic region. The system that has been scrutinized in this case, is a large banking system, being used by multiple banks and financial institutions, serving a large number of users, initiating a very large number of transactions on a regular basis with known peak periods such as e.g. paycheck days, Christmas shopping, etc. The system is naturally highly business critical where performance, and hereof scalability is a fundamental quality parameter. Consequently, scalability debt is of high relevance in this domain and can have a large impact.

The system can be seen as a typical three-tier solution, with external interfaces – typically to banks, a middle-layer offering services for payments, credit cards, fraud detection, etc. The lower tier is the core system, holding all accounts and transaction history.

The usage patterns and usage frequencies of the system is well known today, where new and unknown use via TPPs being identified as a major challenge that needs to be better understood and controlled. Hence, the need for a fast approach to identify potential scalability debt.

V. Method

BankTech started a collaboration with the authors in May 2018. At that time, it was evident that a better understanding was required of the potential impact on existing systems and that a too high increase in requests via TPPs could challenge the scalability and potentially become an unacceptable risk.

Given the tight timeline, a lightweight and fast risk analysis was needed to better identify scalability debt and its impact emerging from an increased external load to prepare for the potential increase.

We describe the method that was developed and used together with BankTech as a novel lightweight approach to identify scalability debt. The method consists of an analysis carried out in 3 stages, explained in the following sections. The descriptions are kept high-level to make them applicable to other domains and cases.

VI. Stage 1: Scalability triage

Scalability debt needs to be identified, estimated and prioritized. We found that the best way forward was to prioritize first the areas of the system that would more likely cause a large impact if affected by scalability debt due to increased load from external TPPs. The first step would therefore limit the in-depth analysis of debt and impact to a prioritized set of operations, instead of analyzing the whole system and prioritizing the outcome afterwards, which could turn out to be very time consuming. For this first step, we used a scalability triage.

A scalability triage is an expert group meeting with the objective of identifying vulnerable/critical parts of the system [4]. The concept of a triage is borrowed from emergency medicine where a doctor quickly determines whether a person requires immediate treatment or can wait. In the same manner – for an operation of a system – it becomes critical if at least one of the following conditions is true:

1. The operation affects functionality, which already has a scalability risk.
2. The operation invokes considerable processing, storage, or communication.
3. We may coarsely classify work and load as small, medium, and large, and quality thresholds as loose, medium, or tough. If the relation between them seems nontrivial, we have a scalability risk.

In this informal step, work, load, and quality thresholds are not specified in detail but are deliberately kept high-level to enable discussions among experts. Therefore, considerable expertise is required.

A team of two researchers and three core roles at BankTech (solution architect, head of non-functional testing, and responsible for system development processes) analyzed an overall architectural model of the system showing existing and new (TPP) interfaces, various components such as fraud detection, payments, and the core system, holding the updated account information. The model also showed new interface components that are needed to manage PSD2.

We started with a high-level architectural diagram showing the main parts of the system, their components, and how they interact. Based on this internal document, we quickly identified a set of 10 potential operations that may be initiated by TPPs. Each operation was described in the term of work, load, and response threshold. Work and load were classified either as Low (L), Medium (M) or High (H). These values were identified collectively by a group of experts. Diverging views were resolved through discussions, leading to collective expert judgements [7].

Through a scalability triage, estimates were made based on expert judgements from the internal core roles. Operations with Work and Load set as H/H were classified as critical. The team identified three such operations. The reduction from ten to three operations may seem minor, but in practice, this reduction turned the problem of identifying debt and impact by finding more detailed requirements from an overwhelming exercise into a manageable task. For the three critical operations, we conducted a more comprehensive analysis using the scenarios described in the next section.

VII. Stage 2: Addressing Affected Subsystems

Once the three operations that were more likely to contain dangerous scalability debt were prioritized, we proceeded to localize the debt with respect to the affected sub-systems. Several internal subsystems internal to BankTech will be affected by PSD2. After speaking to representatives from four of these systems, we got the following basic insights:

(1) They wait for the PSD2-project to give them requirements. (2) They expect the initial workload from PSD2 to be minor, so that they have time to adjust. (3) Some of the PSD2 workload will simply replace existing workload. (4) For other PSD2 operations, the workload will increase. (5) PSD2 workload is uncertain, and to be prepared, they assumed strict scalability requirements.

As a result, we had to get closer in finding the requirements, as described in the next section.

VIII. Stage 3: Estimating impact

The third step was to estimate the impact of the scalability debt. First of all, if the architecture does not support the scalability requirements as anticipated, then the banks may likely incur fines for not complying with the PSD2 requirements. This can generate quite a lot of interest (extra-costs) because of the identified scalability debt. The impact was calculated with different scenarios in mind, as explained below.

To build a better understanding, BankTech and the researchers made a simple estimation model in the form of a spreadsheet containing generic and operation-specific variables. Examples of generic variables are number of TPP apps in use per customer, number of accounts in use per customer, etc. Then – for each of the three critical operations that were identified – specific estimates for burstiness and maximum requests were made. These estimates were based on real BankTech traffic data. These generic and specific input parameters were then used to calculate expected load for each operation.

The workload might increase in three ways:

1. More banks as customers, which gives a corresponding increase in load. BankTech has quite good overview over this.
2. New TPPs give increased workload on the system of BankTechs clients. This is harder to estimate.
3. The operations become more demanding, leading to more work. This was not analyzed.

As the uncertainty is very high – but with potentially severe consequences, the impact estimation describes three scenarios; 'realistic', 'possible', and 'extreme':

- Realistic: This reflects the current customer base. When the predicted number of end-users of BankTech's customers start to use TPP solutions, then we will reach the realistic scenario. This is likely to happen approximately one year after September 2019, when PSD2 should be in operation.
- Possible: BankTech gets more banks as customers, and in addition one of the large players, e.g. Facebook, becomes a

TPP. Also, a larger part of the customers use TPP solutions.

- Extreme: BankTech's banks get two of the big players in the market as TPPs. In addition, even more of the customers will use the TPP solutions.

Our understanding of the existing scalability debt was developed through a series of 17 short meetings with a variety of roles in BankTechs organization. The spreadsheet reflecting this understanding became a very efficient tool to guide discussions, balance viewpoints, and immediately show estimates. The evolving spreadsheet also made it possible to recruit new experts as they found it useful. In the later meetings, the model sparked discussions on what could be acceptable levels regarding scalability and performance. Based on the parameters in the spreadsheet, we estimated load on the three critical operation for the three different scenarios. We used coloring to visualize the impact of scalability debt:

- Green: no or low impact (interest) is estimated to be generated because of scalability debt
- Yellow: some impact is caused by existing scalability debt; something which should be investigated further.
- Red: large impact due to scalability debt; something which now was outside of the capabilities of the system, but which could be possible to handle with careful consideration and good planning.

The resulting spreadsheet was used as a lightweight documentation of the existing scalability debt and its interest to be paid, caused by the potential increase in new external load.

IX. Discussions

In this section, we will discuss the process we have done and the result we have obtained. We also try to flesh out what can be considered more generalizable than the specific case analyzed here.

A. *The need of a lightweight approach*

Based on the open bank case, we gather the following reflections:

1. Altogether, we have now spent 32 expert BankTech hours in 17 meetings, so that a bit less than two BankTech experts attended each meeting, lasting approximately one hour. This is therefore, a light-weight and fast technique. Keeping it light-weight makes it easier to get access to key personnel and to organize more meetings to evolve the analysis.
2. Initially, we focused on prioritizing the areas that were more prone to scalability debt by getting a deep understanding together with one BankTech expert. Later, we involved more colleagues, before we lately have had meetings with the affected sub systems. In this way, we started out with a few internal persons from the case, then developed the simple model, which got attention from others, which then was involved to detail estimates. Through this snowballing-line approach, we have at least ensured that the key roles and experts have been involved and that experience covering most aspects of the solution have been captured and balanced. This approach enabled us to identify key personnel and experts along the way, which was initially hard to identify. We also saw that the analysis gained more attention and traction over time, especially when we could trace who had been involved in the organization.

Although we do not have data yet to validate the precision of the estimates and priorities that were made, we do see that such a light-weight process gained attention in the organization from the easy start with one person. After a short period of time, the results are now being used by BankTech to guide their preparations to manage scalability when PSD2 comes into effect.

B. *Other aspects of ATD*

In this paper, we report a method related to scalability triage, a lightweight technique assisting primarily on identification, estimation and prioritization of ATD in the form of scalability debt. Other aspects of ATD are relevant, although not covered in this paper, and should be addressed in future studies:

- Measurement: to gather additional evidence related to the scalability debt, we are about to compare expected scalability requirements with actual capabilities.
- Monitoring: the approach described here can be done in an iterative fashion, following Agile principles. We are working with the integration of this method with agile practices [4].
- Prioritization: the scalability triage described here highlights, which parts of the system have the most risk of impact due to scalability debt, which helps prioritizing the parts of the systems to be analyzed and refactored. However, technical debt prioritization should also involve the comparison and ranking of scalability debt with respect to the development of other features or other types of ATD.
- Repayment: the cost of refactoring scalability debt is also an important parameter to be studied to understand if scalability debt should be fixed or not.
- Communication and documentation: in this approach, the main scalability debt documentation was a spreadsheet. However, the documentation may be improved with the design of a better tool which would make the triage and its iteration more efficient.

C. *Open Systems generate unpredictable (scalability) debt*

Digitalization brings new challenges for software development, with the transformation of previously independent applications with locked in data and business processes. "Open systems" are part of a complex ecosystem and therefore cannot be considered isolated anymore. Requirements are shared from many different parties, and the presence of a collection of heterogeneous applications makes it hard to plan such a complex landscape of independent applications upfront. For technical debt, this is a new challenge, and it might become increasingly difficult to avoid it with upfront design decisions. Open banking is a large example, but there are many more, for example, open systems based on transport data.

During design, functionalities and features get most attention. However, in this case we show how analysis of accumulating technical debt (in this case related to scalability) is the key to avoid a huge impact in the future. With our scalability triage technique, we report a practical approach that supports the management of scalability debt. In addition, our method is lightweight, although based on expert opinion. In the investigated case, the method was used to do an analysis over a discrete period, but may also be part of a continuous development process.

X. State of the Art

Architectural debt is regarded as sub-optimal architectural solutions [8]. In practice, there is ATD when the architectural tradeoff among qualities is not optimal with respect to business goals. A holistic method to tackle ATD was proposed by [9] by taking in consideration several factors affected by the ATD, such as development speed, defects and other qualities. However, in such a holistic method, scalability has been taken in consideration as a high-level factor with an overall score, without proposing an in-depth analysis. ATD is well known to affect maintainability and evolvability of a system, as reported in the official definition of technical debt [10]. Also, several studies are related to the study of ATD with respect to structural issues of the code. A quantification for such issues and their impact on maintainability and evolvability is reported via the study structural metrics and their negative impact (e.g. [11]). However, recent research suggests that other qualities might be affected by ATD, such as performance [12] reusability, etc. [13]. In addition, scalability has been found to be a critical quality to be addressed when removing technical debt before the growth of startups [14]. Declining scalability is reported as a sign of technical debt [15]. In conclusion, although scalability is considered a key quality affected by technical debt, scalability debt has not been studied explicitly.

A recent study focused on taking scalability as input to compute the interest of Technical Debt in a cloud-based architecture using real options [6]. Such an approach is based on architects monitoring a service and proposing a change in the architecture based on a suboptimal solution such as the complexity of such service. Then, the approach computes if such architectural change is convenient according to value and cost of refactoring in different paths. In our study, we do not necessarily refer to cloud-based applications, and we propose to start from the analysis of the most critical operations for the business, which aims at identifying and prioritizing the ATD with respect to the risk of paying the interest. The subsequent evaluation of the ATD has not been studied yet, and could involve an approach such as the one proposed by the authors in [6] if suitable.

XI. Conclusions and further work

In this paper, we have studied architectural technical debt specifically related to scalability, which we call scalability debt. We presented a case related to open systems (open banking), where uncertainty causes scalability debt to be acquired continuously. We argue that, in such systems, a continuous and lightweight approach to manage scalability debt is required. We report a lightweight method used in an open banking organization to identify, estimate and prioritize scalability debt. Such method consists of three phases, an initial triage to prioritize the areas with more likely impact due to scalability debt, a second phase where the debt is assessed in different affected subsystems and a third phase where the impact (interest) of the debt is calculated according to three scenarios. Different thresholds and colors were used to document and communicate the scalability debt and its impact to stakeholders. A feature of the approach is the way it involved experts in the organization, starting with a few core experts, attracting more attention and more experts and key roles as the analysis grows and the risks are being identified. The results can be taken into consideration by practitioners when dealing with scalability debt and may be used as starting point to develop the method presented here further or to research novel lightweight approaches to manage technical debt related to other key business qualities.

Acknowledgments

The research leading to these results has received funding from the Norwegian Research Council under grant #256669 (ScrumScale). We thank BankTech for a generous cooperation leading to the insights shared in this paper. In respect for the business critically of scalability debt in BankTech, exact numbers for scalability requirements are not shared in this paper.

References

[1] A. Martini, J. Bosch, and M. Chaudron, "Investigating Architectural Technical Debt accumulation and refactoring over time: A multiple-case study," *Information and Software Technology,* vol. 67, pp. 237-253, 2015.

[2] T. Besker, A. Martini, and J. Bosch, "Managing architectural technical debt: A unified model and systematic literature review," *Journal of Systems and Software,* vol. 135, pp. 1-16, 2018.

[3] S. Becker, G. Brataas, and S. Lehrig, *Engineering Scalable, Elastic, and Cost-Efficient Cloud Computing Applications: The CloudScale Method*. Springer, 2017.

[4] G. Brataas, G. K. Hanssen, and G. Ræder, "Towards Agile Scalability Engineering," in *International Conference on Agile Software Development*, 2018, pp. 248-255: Springer.

[5] G. Brataas, N. Herbst, S. Ivansek, and J. Polutnik, "Scalability Analysis of Cloud Software Services," in *Autonomic Computing (ICAC), 2017 IEEE International Conference on*, 2017, pp. 285-292.

[6] E. Alzaghoul and R. Bahsoon, "Evaluating technical debt in cloud-based architectures using real options," in *2014 23rd Australian Software Engineering Conference*, 2014, pp. 1-10: IEEE.

[7] M. Jørgensen, B. Boehm, and S. Rifkin, "Software development effort estimation: Formal models or expert judgment?," *IEEE software,* vol. 26, no. 2, pp. 14-19, 2009.

[8] Z. Li, P. Avgeriou, and P. Liang, "A systematic mapping study on technical debt and its management," *Journal of Systems and Software,* vol. 101, pp. 193-220, 2015.

[9] A. Martini and J. Bosch, "An empirically developed method to aid decisions on architectural technical debt refactoring: AnaConDebt," in *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, 2016, pp. 31-40: IEEE.

[10] P. Avgeriou, P. Kruchten, I. Ozkaya, and C. Seaman, "Managing technical debt in software engineering (dagstuhl seminar 16162)," in *Dagstuhl Reports*, 2016, vol. 6, no. 4: Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[11] R. Kazman *et al.*, "A case study in locating the architectural roots of technical debt," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, 2015, vol. 2, pp. 179-188: IEEE.

[12] U. Eliasson, A. Martini, R. Kaufmann, and S. Odeh, "Identifying and visualizing Architectural Debt and its efficiency interest in the automotive domain: A case study," in *2015 IEEE 7th International Workshop on Managing Technical Debt (MTD)*, 2015, pp. 33-40: IEEE.

[13] T. Besker, A. Martini, and J. Bosch, "Time to Pay Up: Technical Debt from a Software Quality Perspective," in *CIbSE*, 2017, pp. 235-248.

[14] T. Besker, A. Martini, R. E. Lokuge, K. Blincoe, and J. Bosch, "Embracing Technical Debt, from a Startup Company Perspective," in *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2018, pp. 415-425: IEEE.

[15] E. Tom, A. Aurum, and R. Vidgen, "An exploration of technical debt," *Journal of Systems and Software,* vol. 86, no. 6, pp. 1498-1516, 2013.