Exploring technical debt of database architecture evolution

RuJia Li, *Student Member, IEEE*

Abstract—Technical debt is trading long-term software quality for short-term benefits. The metaphor has been used to characterize and quantify issues arising from suboptimal technical software evolution decisions that are not long-term focused. Distributed database tend to underlie many large scale software. However, their debts are not widely studied. In this paper, we draw on a case study to analyse the link between technical debts in distributed database architectures and representative architecture evolution decisions. We use alibaba's for this purpose.

Index Terms—Technical debt, distributed database,architecture evolution.

1 Introduction

TECHNICAL debt is a situation in which long-term code quality is traded for short-term gain [2] . Quick and poor quality in incepting the architecture may give you short term gains but can incur long interest-like payment. This implies that one may need to pay an extra effort in future development or maintenance. Improving quality can come with extra cost and effort. We use technical debt to quantify the value of evolution decisions in database. We trade developing decisions and their value against payback strategies

There has been many research aimed at studying technical debt, some of which are theortical and other are experimental. However, fewer research has used case studies to evaluate, analyse and model technical debt. In this paper we use alibaba's example to explore the relationship between technical debt and architectures evolution decisions.

The rest of paper is organized as follows: we first briefly introduce technical debt and alibaba database evolution; then we explore the deepest driving force of these evolution from productivity and future usability; in the third section a dynamic approach was introduced for modelling and quantifying the technical debt. Lastly, we close the paper with conclusions and a discussion of future work.

2 Overview of Technical Debt

Technical debt a concept in programming that reflects the extra development work that arises when code that is easy to implement in the short run is used instead of applying the best overall solution [1]. If technical debt is not repaid, it can accumulate 'interest', making it harder to implement changes later on [3]. Unaddressed technical debt increases software entropy [4]

- *RJ.Li is with School of Computer Science,University of Birmingham, Birmingham, UK, B15 2TT. E-mail: rxl635@cs.bham.ac.uk*
- *R. Bahsoon and P. Hancox are with University of Birmingham.*

Manuscript received April 19, 2016; revised August 26, 2016.

3 Alibaba and its Database Architecture Evolution

Alibaba Group is a global leader in online commerce [5]. Hundreds of millions of users, merchants and businesses accessed three commercial websites such as Taobao,Tmall and Alibaba.com.

From the official data, in 2016 the peak daily turnover is over 12 billion. The peak daily data increment is 200TB approximately [5]. In the last 15 years,database architecture of Alibaba has experienced four time changes. Form 2003 to 2004,they used simple MySql clustering. From 2005 to 2010,they used Oracle dataguard. From 2011 to 2015,they used Self-developed database AliSql. In fact, it is one branch of MySQL.Currently, they are using Self-developed database OceanBase. I will describe them briefly.....

3.0.1 stage1 MySQL

Mysql master/slave structure was used in the initial stage of the alibaba. This architecture enables data transfer from one master database to one or more slave database servers automatically, the master database server is responsible for writing,and the slave database servers is responsible for reading.

3.0.2 stage2 Oracle

From a security point of view, alibaba's decision makers gave up the Mysql database and started to use Oracle in 2005. To protect Oracle data from disasters, data corruptions and human error,data guard was introduced. Oracle's data guard provides the feature of synchronizing data copies.

3.0.3 stage3 AliSQL

Due to the expensive price of Oracle service and the limit of single machine upgrade,Alibaba decided to research and development AliSql database. they improved the performance and functionality based on Mysql to meet the alibaba business's needs.

3.0.4 stage4 OceanBase

OceanBase is a distributed and extensible relational database, It owns many features such as SQL interface, scalability, continuous availability, etc.There have been plenty of instances in Alibaba and serves tens of billions read and write transactions every day [7].

4 Reasons of database architecture evolution

As with any other artifact produced as part of the software life cycle [8], database architectures must be modified to cope with changing requirements. Such requirements can relate to fixing poor secure design, scaling for data, changes in robustness requirements etc. In order to adapt to these requirements, changes may incur interest due to previous quick and dirty database design decisions. It is possible for stakeholders to continue paying the interest or abandon the quick and dirty design. So what determines architecture evolution? In order to answer this question, architecture productivity was introduced.

Productivity is determined by the input of an activity and its output [9]. Therefore to measure architecture productivity ,the input and output of architecture must be measured. Unfortunately, there is no standard output for architecture, the next simulation experiment is based on the fact that implementing the same features under the same quality. For example, supporting 100 thousand concurrent transactions, getting the result by calculating different man/day.

Two scenarios will be supposed. In scenario one, the stakeholders continue using old database architecture which have a higher productivity now, but the productivity of this architecture decreases quickly due to technical debt. in scenario two, stakeholders decide to apply a new database architecture for reducing interest in the future. Initially,it is clear that the cost of scenario two is much higher than scenario one's, when achieving the same features because of reconstruction needs more effort.

TABLE 1: simulation experiment for long-term productivity

Scenario/Time	year1	year2	year3	year4	year5
one	30	50	60	90	120
two	120	60	40	35	25

The simulation results are shown in table 1. In first and second year scenario one have a lower cost in terms of achieving the same function. However, it increases dramatically because code deteriorates and harder-modify. On the contrary, the cost of scenario two is very high at first, but after 5 years it become fewer due to good design and plan. Even if it is only a hypothesis,we can not proof that this effect occurs but many of us feel that this explains what we see architecture qualitatively [9].

These scenarios indicate an trade-off between short-term and long-term productivity in the architecture development cycle. Evidently, one of important reasons for database architecture evolution is to get a long-term productivity. Also, another reason is that the expected architecture of database can not catch up the business trend in the future.

Alibabas history was described for explaining this view. In 2000, Alibaba has built the first generation database MySql Culter. Alibaba was expected to run the database for a long period for about 8 years. In fact, they began to look for new alternatives in the third years.

As it shown in figure 1, we assume that commercial development is linear (ideal state). Technical debt affects the architecture design, increases maintenance cost and decreases development efficiency, Ultimately, it accelerates the change of database architecture and made the expected architecture can not catch up future market trend.

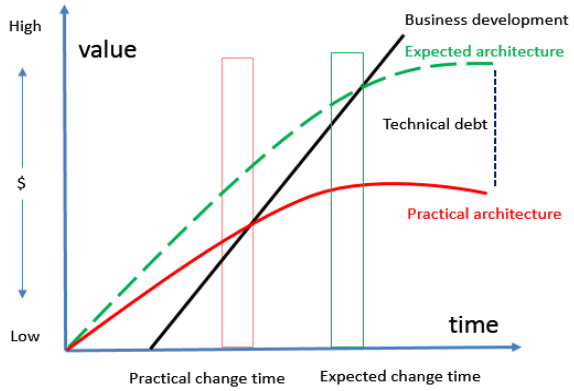

Fig. 1: relationship between evolution time and technical debt cost

In reality, the situation will be more difficult than mentioned, because we can not determine the development of business. Therefore, as a decision maker, they should pay more attention to the time factor when designing the architecture.

In conclusion, the reasons we need to change architecture continually are the poor productivity of future architecture and the inability of the expected architecture to catch up future market trend.

5 Approach of modelling database architecture evolution

Based on Tom DeMarcos theory "You cant control what you can't measure." [10]. When it comes to measure technical debt software. Many methods for technical debt identification, quantification and management have been proposed. However, most of them are deterministic model and based on coding debt, which is not suitable for our database architecture evolution. Database architecture evolution is a dynamic process, using a static search method is difficult to get a optimal solutions.

In this part, we put our eye on technical debt of database architecture evolution and come up a new dynamic analysis method to evaluate it. Also, we use alibaba's database architecture evolution as an example to proof it. The method contains the following steps:

1) Target determination and decomposition;
2) Map sub-target into database metrics;
3) Dynamic technical debt assessment;
4) Multi-criteria technical debt analysis;

The detail is as follow...

5.1 Target determination and decomposition

It is complex to evolve architecture, a careful consideration is needed before any evolution decision has been made. Like any other important business decision, at first an appropriate goal must be identified. An appropriate target means that target can be decomposed. For example, in 2003 with the development of internet and banking business, Alibaba group wanted to release the new product Alipay. Alipay is a third-party online payment platform which highly related to buyer and the seller's money. This target cloud spilt into three parts from the aspect of system level, which includes better transaction, more secure for data, excellent disaster recovery function.

Fig. 2: target decomposition of Alipay

5.2 Map sub-target into database metrics

It is difficult to quantify the target, but we can quantify the system metrics that correspond to the target. As we have mentioned, each large goal can be broken down into many specific small targets. Each small target can be mapped to a different system indicators. How to estimate the distributed database's indicators? we give a example as how to estimate security indicators of alibaba's four databases . As it shown in table 2, We use plus "+" to show this part's estimated result,the more plus "+" it has,the better security indicators it owns. the final result is weighted average of every part's score. It also reflects the different security index for four different database.

5.3 Dynamic technical debt assessment

As we have mentioned in part two, architectural goals can be broken down into small achievable goals, each achievable goal can be mapped to different database indicators. Evaluating the overall technical debt of the architecture evolution is in fact assessing the technical debt raised by each system indicator. However, it is difficult to assess each system indicator's technical debt due to it is changing all the time. We defined technical debt of one indicator at point time as $SampleDTV_i$. Suppose distributed database is static, $SampleDTV_i$ can be calculate easily. However, in the reality, every moment the technical debt will fluctuate from plenty of influence factors such as environmental needs, employee turnover and machine wear. Because of this fluctuation, any given $SampleDTV_i$ value may be atypical. In order to estimate a typical technical debt, therefore it is natural to take some sort of average of the $SampleDTV_i$,we maintains an average, called $EstimatedDTV$, upon calculate a new $SampleDTV_i$, we updates $EstimatedDTV$ according to the following formula:

$$EstimatedDTV_i = (1-\mu)*EstimatedDTV_i+\mu*SampleDTV_i$$

$EstimatedDTV$ is a weighted combination of the previous assessment value of EstimatedDTV and the new assessment value for $SampleDTV_i$. The recommended value of μ is defined by the measure frequency and business factors. Usually, the new $SampleDTV_i$ contribution rate is 0.2 due to minimize the deviation. It means the formula will show as fellow:

$$EstimatedDTV_i = 0.8*EstimatedDTV_i+0.2*SampleDTV_i$$

In conclusion, technical debt assessment is a constantly changing progress,$EstimatedDTV_i$is based by previous assessment and current assessment.

5.4 Multi-criteria technical debt analysis

Up to present, we have known what is the expected database metrics and current database metrics. Also, we have known how to estimate each database metrics's technical debt. The next step is compared to the technical debt of target database metrics and the current metrics. A multi-criteria technical debt analysis method was proposed to find whether it is suitable for evolution for Specified indicator. We use dynamic technical debt method to calculate the technical debt of implementing every metric $EstimatedDTV_i$. It must be point out that each indicator is not isolated. When it comes technical debt of implementing metric α, other metrics's influence must be considered. We propose every metrics have different factorα_i, and the other metrics's technical debt is $oEstimatedDTV_i$, so technical debt of a final metrics $EstimatedDTV_i$ calculated as this formula:

$$EstimatedDTV_i = EstimatedDTV_i+\alpha_i*oEstimatedDTV_i$$

then,we assess achieving metrics value $MetricsVlaue_i$. And by comparing the $EstimatedDTV_i$ and $MetricsVlaue_i$ to decide whether suitable for changing database in terms of this metric. Lastly we make a trade off between different metric.

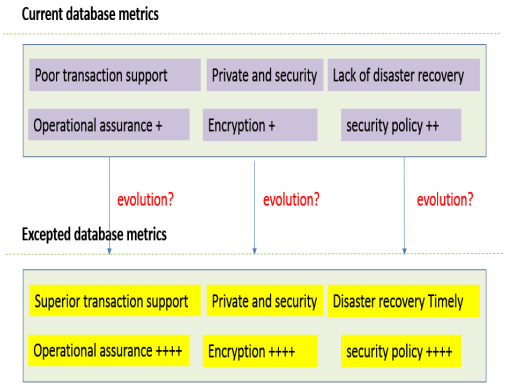

Fig. 3: multi-criteria technical debt analysis

TABLE 2: distributed database security index

safety assessment index	index details	stage1:MySql	stage2:Oracle	stage3:AliSql	stage4:OceanBase
security policy	discretionary access control	+++	++++	+++	+++
	object reuse	++	+++	++	+++
	labels	+	+++	+++	+++
	mandatory access control	++	+++	+++	++
accountability	identification and authentication	+++	++++	+++	+++
	audit	+	++++	+++	++++
operational assurance	system architecture	++	+++	+++	+++
	system integrity	+	++++	+++	+++
	covert channel analysis	++	++++	+++	+++
	trusted facility management	+	+++	+++	+++
	trusted recovery	+	++++	+++	++++
	transaction error rate	+	+++	+++	+++
life cycle assurance	security testing	++++	++++	++++	++++
	design specification and verification	++++	++++	++++	++++
	configuration management	+++	++++	+++	++
	trusted distribution	++	++++	+++	+++
Encryption	key storage	+	+++	++	++
	encryption granularity	++	++++	+++	++
	prevention of information leakage	+	++++	+++	+++
	detection of unauthorized modifications	++	++++	++++	++
documentation	security features user's guide	++++	++++	++++	++
	trusted facility manual	++++	++++	++++	++++
	test documentation	++++	++++	++++	++++
	design documentation	++++	++++	++++	+++
summary		++	++++	+++	+++

6 Conclusion and future work

This paper described Alibaba's four database evolution stages briefly. Then, we took these evolutions as an example to explore such problems: the first one is reasons behind database evolution, during the progress two reason was found, which is poor productivity of future architecture and inability of the expected architecture to catch up future market trend; the second is the approach of modelling database architecture evolution,we use four steps dynamic to assess the technical debt which evolution faced, In order to optimize the decision-making and make profit maximization.

Future work is include that to study every metric value $MetricsVlaue_i$ in detail,since it is an important factor in deciding architecture evolution and to find how to make a trade off of between different metric.

Acknowledgments

The authors would like to thank Dr. Rami Bahsoonfor his participation in this study. We also extend our thanks to Dr. Peter Hancox for his guiding of research skills.

References

[1] Techopedia.com. (2016). What is Technical Debt? - Definition from Techopedia. [online] Available at: https://www.techopedia.com/definition/27913/technical-debt [Accessed 9 Dec. 2016].

[2] Persson, A. and Stirna, J. (2015). Advanced Information Systems Engineering Workshops. In: CAiSE 2015 International Workshops. p.215.

[3] Tom, E., Aurum, A. and Vidgen, R. (2013). An exploration of technical debt. Journal of Systems and Software, 86(6), pp.1498-1516.

[4] En.wikipedia.org. (2016). Technical debt. [online] Available at: https://en.wikipedia.org/wiki?curid=11885652 [Accessed 9 Dec. 2016].

[5] Alibabagroup.com. (2016). Alibaba Group. [online] Available at: http://www.alibabagroup.com/en/about/history [Accessed 8 Dec. 2016].

[6] Fowler, M. (2016). bliki: TechnicalDebtQuadrant. [online] martinfowler.com. Available at: http://www.martinfowler.com/bliki/TechnicalDebtQuadrant.html [Accessed 7 Dec. 2016].

[7] Gao, Y., Shim, K., Ding, Z., Jin, P., Zujie, R., Xiao, Y., Liu, A. and Qiao, S. (2013). Web-Age information management. 1st ed. Berlin: Springer, p.3.

[8] Horkoff, J., Li, T., Li, F., Salnitri, M., Cardoso, E., Giorgini, P., Mylopoulos, J. and Pimentel, J. (2014). Taking goal models downstream: A systematic roadmap. In: 2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS). IEEE.

[9] Fowler, M. (2016). bliki: CannotMeasureProductivity. [online] martinfowler.com. Available at: http://martinfowler.com/bliki/CannotMeasureProductivity.html [Accessed 8 Dec. 2016].

[10] DeMarco, T. (1982). Controlling software projects. Management, measurement and estimation. 1st ed. New York,N.Y.,Yourdon,1982,: Prentice Hall;, p.3.

RuJia Li received the bachelor degree in computer science from Wuhan University for his research on studying asynchronous non-blocking Server,In the same year,He got another bachelor degree in business management,After that he went on working in the largest electric company ”State grid corporation of China ”, Now, he is studying advanced computer science in University of Birmingham, Birmingham,United Kingdom.He is a oracle certified professional and student IEEE member .