On the Benefits of the Accelerate Metrics: An Industrial Survey at Vendasta

Francesco Lomio
Tampere University
francesco.lomio@tuni.fi

Zadia Codabux
University of Saskatchewan
zadiacodabux@ieee.org

Dale Birtch
Vendasta
dbirtch@vendasta.com

Dale Hopkins
Vendasta
dhopkins@vendasta.com

Davide Taibi
Tampere University
davide.taibi@tuni.fi

***Abstract*—The popularity of the Accelerate metrics is increasing in the industry. The Accelerate metrics are four key metrics to evaluate the software delivery performance: lead time for changes, deployment frequency, mean time to recover, change fail rate. However, their benefits in monitoring the development process performance of microservice-based systems have not been evaluated. In this study, we analyze the case of Vendasta, a Canadian company that migrated to microservices two years ago and adopted the Accelerate metrics to monitor their development process. Our goal is to understand whether these metrics are beneficial in the microservices context from the practitioners' point of view. Therefore, we surveyed employees from different teams and obtained 62 responses. Our results show that the Accelerate metrics provide a good overview of the process issues and are particularly helpful for a high-level representation of the process performances. Furthermore, the Accelerate metrics also enabled the teams to improve their productivity, significantly reducing service outages.**

***Index Terms*—Accelerate Metrics, Microservices, Survey, Technical Debt, Four Key Metrics, FKM, DevOps, Lead Time for Changes, Deployment Frequency, Mean Time to Recover, Change Fail Rate.**

I. Introduction

A plethora of metrics have been used to measure various aspects of software development using Agile practices. Typical metrics are burndown (amount of work done over time), velocity (team's completed estimates over time), lead time (time between start and end of a task), and bug counts [1]. Kupiainen et al. [2] also reported a comprehensive list of metrics such as business value delivered, customer satisfaction, number of test cases, technical debt in effort, effort estimate, among others, which are categorized based on their application.

A few years ago, Thoughtworks proposed four key metrics called the Accelerate metrics to measure software delivery performance [3]. The proposed metrics are the following:

- Lead Time for Changes - the average time taken by a commit to go from deployment to production
- Deployment Frequency - how often are the services successfully released to production
- Mean Time to Recover - the time taken to recover from a failure in production
- Change Fail Rate - the percentage of deployments causing a failure in production

Despite these metrics being similar to those used in Agile software development, it is still unclear how they could benefit microservices-based systems. Microservices are a set of single services, each deployed independently and with a single purpose[1]. They enable the decrease of communication overhead between teams, improve team's velocity, create a more understandable system, increase team's independence, and reduce its technical debt [4]–[7]. With microservices, each team can work and deploy independently without synchronizing with other teams. However, the activities related to managing and keeping track of the development process, and in particular, monitoring the productivity, increase significantly [5], [8]. Since metrics used in monolithic systems and global software development often do not translate well to microservices [9], Accelerate metrics could be a good proxy to get an overview of the process performance also in the case of microservice-based systems [3].

To the best of our knowledge, little to no studies have investigated the benefit of the Accelerate metrics. Sallin et al. [10] analyzed the application of the Accelerate metrics in measuring the software delivery performance by automatically calculating the metrics and showing them in production to a team of practitioners.

Vendasta[2], a Canadian company that specialized in providing Enterprise Resource Planning software solutions (e.g. sales, marketing, finance, etc.) to small businesses, migrated to microservices two years ago and adopted the Accelerate metrics to keep better track of the development process of its services. The choice of these metrics was motivated by the need to provide a set of measurements for developers to understand where to improve their processes.

Our study aims to evaluate the benefits of the Accelerate metrics in the context of microservices at Vendasta. Therefore, we surveyed its development teams and analyzed the answers to understand the developers' perception regarding the benefits and usefulness of the Accelerate metrics to their services. The outcome of this study will be helpful for researchers to understand the impact of the Accelerate metrics better. Moreover, it will help practitioners understand how to incorporate the

This study is partly supported by the Natural Sciences and Engineering Research Council of Canada (RGPIN-2021-04232 and DGECR-2021-00283) at the University of Saskatchewan.

[1]https://www.martinfowler.com/articles/microservices.html
[2]https://www.vendasta.com/

Accelerate metrics for monitoring purposes during software development.

The remainder of this paper is structured as follow. Section II describes the methodology. In Section III, the results are presented and are further discussed in Section IV. Threats to validity is listed in Section V. Finally, Section VI presents the conclusion and highlights the future works.

II. Methodology

A. Goal and Research Questions

Our goal is to investigate the benefits and usefulness of the Accelerate metrics in the context of the microservices-based system at Vendasta. Therefore, we derived the following Research Questions (RQs).

RQ1: How does technical debt impact the development and delivery?

RQ2: What is the perceived benefit of the Accelerate metrics?

RQ3: What is the perceived usefulness of the Accelerate metrics?

RQ1 investigates the presence and impact of technical debt on various aspects of the services. In an ideal system, with no technical debt nor defects and hence no problems or failure, the usage of these metrics would be different, if not useless. For instance, the Mean Time to Recover or Change Fail Rate would be null. RQ1 is of utmost importance for the continuation of this study. If the system is debt- and defect-free, the other RQs would be superfluous. In RQ2 and RQ3, we aim to identify and evaluate the benefits and usefulness of using the Accelerate metrics.

To answer the RQs, we designed a survey comprising of 20 questions, both closed- and open-ended. In Table I, we provide a summarized version of the survey with the respective RQs. The closed questions are either Likert scale or numerical. A 5-points Likert scale has been used, with the points being: (1) Highly Disagree; (2) Disagree; (3) Neither Agree nor Disagree; (4) Agree; (5) Highly Agree. We also have a set of questions to collect demographic information of the participants. First, the participants were asked how the technical debt impacts the development and delivery and the sacrifices needed to respect the delivery timetables. They were also asked an estimate of the time needed for removing all the technical debt accumulated. All these questions were asked after the definition of technical debt was given as "the implied cost of additional rework caused by choosing an easy (limited) solution now instead of using a better approach that would take longer." Next, the participants were asked about the benefits they perceived using the different Accelerate metrics in developing and delivering the services they work on. Finally, we investigated the usefulness of the Accelerate metrics and how their adoption affects the development and performance of the developers.

The survey was rolled out in two phases. It was initially distributed among a few employees (five) holding different positions. The purpose of this initial roll-out was to fine-tune the questions and remove ambiguities. After the initial phase, the survey was distributed to all the software engineers. The survey was advertised using an internal mailing list and announcements during weekly meetings. It was initially rolled out for six weeks during Summer 2021 and an additional two weeks to allow more responses from those who could not complete it on time. In total, the roll-out lasted 64 days.

B. Data Analysis

The responses to the closed questions were analyzed using descriptive statistics. This approach allowed us to have an overview of the answers and better understand the participants' general opinions. Moreover, we also considered the distribution of the answers given for each point in the Likert scale questions. The open-ended questions were analyzed using emergent coding, i.e., the codes to categorize the data come from the data itself.

C. Ethical Considerations

Given the sensitive nature of the survey, the study was approved by the Ethics Committee of the Tampere Region, with statement number 58/2021.

III. Results

Out of the 110 practitioners invited to the study, 62 participated in the survey. The participants' median years of experience are six, with 17 having more than ten years of professional experience. Moreover, the median number of years spent by the participants at Vendasta is 2.5 years. The participants come from a total of 22 teams. The median number of years in the current team for the practitioners is one year. Each team has an average of four software engineers. Next, we report the results of the RQs.

A. Technical Debt Impact on Development and Delivery (RQ1)

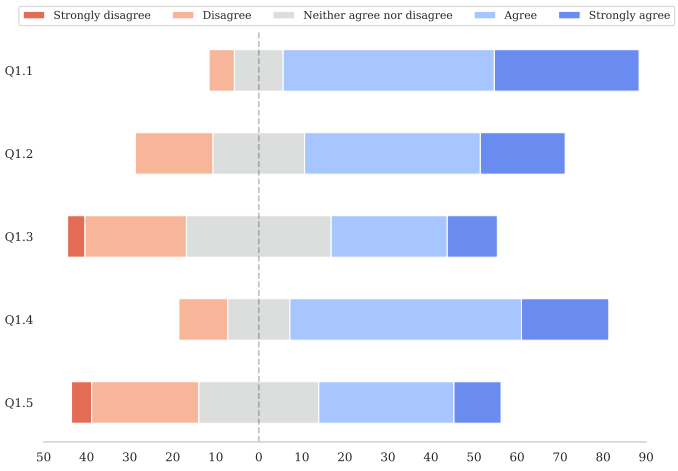

Fig. 1. Technical Debt Impact (RQ1). On the x-axis is presented the frequency of the answers given in percentage: the percentages of respondents who agree with the statement are shown to the right while the percentage of the ones who disagree is on the left of the 0.

The majority of the participants confirmed that the services they worked on accumulated technical debt over time (85%)

TABLE I
LIST OF QUESTIONS

RQ		Question	Type
RQ1		**Technical Debt Impact**	
		We define Technical Debt as the implied cost of additional rework caused by choosing an easy (limited) solution now instead of using a better approach that would take longer	
	Q1.1	My services have accumulated technical debt	L. Scale
	Q1.2	Technical debt of my services impacts the overall speed of delivering my services	L. Scale
	Q1.3	Technical debt of my services dependencies impacts the overall speed of delivering my services	L. Scale
	Q1.4	Technical debt of my services impacts the speed of delivery for new features	L. Scale
	Q1.5	Technical debt of my services dependencies impacts the speed of delivery for new features	L. Scale
	Q1.6	In the past 6 months, which of the following have you sacrificed to move quicker	Multiple Choice
	Q1.7	How many weeks do you think your team would need to completely get rid of all the technical debt for the services you develop? (average weeks per service)	Numerical
	Q1.8	How many services are your service dependent on?	Numerical
RQ2		**Perceived Benefit of the Accelerate Metrics**	
		What are the benefits of having the following information:	
	Q2.1	– the Deployment Frequency (per week) for your service?	Open
	Q2.2	– the Change Fail Rate for your service?	Open
	Q2.3	– the Mean Time to Recover (in hours) for your service?	Open
	Q2.4	– the Average Lead Time (in days) for your service?	Open
RQ3		**Perceived Usefulness of the Accelerate Metrics**	
	Q3.1	Having the Deployment Frequency (per week) metric would improve my performance	L. Scale
	Q3.2	Deployment Frequency is an important metric to me	L. Scale
	Q3.3	Knowing the Change Fail Rate would influence my performance	L. Scale
	Q3.4	Change Fail Rate is an important metric to me	L. Scale
	Q3.5	Knowing the Mean Time to Recover (hours) would influence my performance	L. Scale
	Q3.6	Mean Time to Recover is an important metric to me	L. Scale
	Q3.7	Knowing the Lead Time would influence my performance	L. Scale
	Q3.8	Lead Time is an important metric to me	L. Scale
		Likert scale: 1-Highly Disagree, 2-Disagree, 3-Neither Agree nor Disagree, 4-Agree, 5-Highly Agree	

and confirmed its negative effect on the speed of their delivery (66%). Most of the participants also agreed that technical debt impacts the delivery of the new features (71%). Regarding the impact of technical debt on services' dependencies, the participants generally agreed that it impacted the delivery of new features (45%). These results are depicted in Figure 1.

When asked which activities have been sacrificed in order to move quicker (Q1.6), the majority of participants reported *"refactoring complex code"* (29%), followed by *"incomplete e2e coverage"* (23%), *"incomplete unit tests"* (23%), and *"adding a failing test before a fix" (17%)* as depicted in Figure 2. We also had an option for "others" where the participants could provide additional activities that they sacrificed. Some of the responses include *"gathering customer feedback," "feature development," "interruptions/operational issues," "fixing user problems," and "migrating legacy applications to microservices"*. One participant mentioned that he sacrificed all the activities we mentioned but not to move quicker and did not elaborate any further.

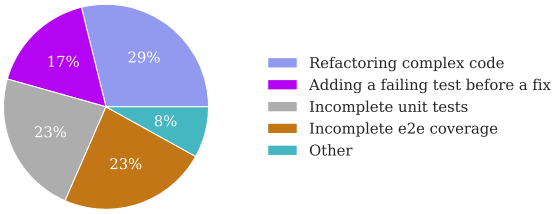

Fig. 2. Sacrificed Actions to Move Quicker (Q1.6)

While we are aware that technical debt cannot be realistically completely eliminated, the participants were asked how much time (in weeks) it would take the team to eliminate all the technical debt for the services they develop (Q1.7). The responses to this varied considerably. 34 of the responses range from one week to three months. Ten of the responses were three to six months. 12 responses were six months to a year. Additional responses include 1.5 years, 20+ years and "too large, cannot be measured." One of the participants pointed out that *"things change so fast that even things done perfectly a month ago could be tech[nical] debt now."* Another participant's response was along these lines *"fixing the technical debt today does not get rid of technical debt in general. If we got rid of all of what we consider TD today and tomorrow there are new business requirements, then we may have more TD."*

It is important to note that each service depends on different services (from one to 14, with an average of 4 service dependencies for each of them). Therefore, services with higher dependencies can also have a higher coupling, and therefore, be more prone to have issues. For instance, even if each service is independent of other services, a service with a strong coupling with another service might need to wait for the other service to deliver, thereby being forced to synchronize its DEPLOYMENT FREQUENCY with the service it depends on.

B. Perceived Benefit (RQ2)

Out of the 62 participants, 58 answered these questions (Q2.1 - Q2.4). The majority of the participants agreed that DEPLOYMENT FREQUENCY allowed teams and developers to

have an overview of the progress and challenges of their services, i.e., it is an indication of whether the release frequency is large or small - with a larger frequency being preferred since it means faster deployment of services. In contrast, a less frequent release is synonymous with some features taking longer and would be helpful to draw attention to potential issues. Also, some of the participants mentioned that the metric could be used to justify and prioritize technical debt cleanup. Despite these benefits, some of the participants (about 20%) reported that the metric was not valuable, especially on its own, but it could become beneficial with the CHANGE FAIL RATE or as a context for the other Accelerate metrics.

The CHANGE FAIL RATE is considered a good indicator of deployment failure, showing a need to "slow down". For some of the participants, it is an indicator of risk, quality and might indicate issues in testing. Also, this metric shows the frequency of roll-backs, and some of the participants found it to be more helpful than DEPLOYMENT FREQUENCY in addressing the technical debt in a service.

About half of the participants agreed that MEAN TIME TO RECOVER could help in understanding how quickly the service can recover from issues in production, therefore indicating the reliability and stability of service to customers. In addition, a participant mentioned that this metric could also help define or improve recovery processes, while another suggested that it could indicate technical debt if the time to recover is unreasonably long.

Regarding LEAD TIME FOR CHANGES, the participants found that it could be beneficial to understand whether user stories and tasks were correctly estimated, and therefore, can be used for future story prediction, and how it can reveal potential issues when tasks or stories are not completed on time. Despite this, around 30% of the participant did not find this metric beneficial, as the delivery time heavily depends on the difficulty of the task.

C. Perceived Usefulness (RQ3)

According to the participants, the most valuable metric is CHANGE FAIL RATE (73%), with the majority of them thinking that knowing this metric would influence their performance (61%). The MEAN TIME TO RECOVER was also deemed helpful by the majority of the participants (66%), with 48% of the participants considering it impactful on their performances. The metric that caused mixed opinions is the LEAD TIME FOR CHANGES, with almost the same number of participants deeming it valuable and not valuable (39% and 35% respectively). Regarding its perceived impact on the performances, 34% of the participants considered it impactful, while 36% did not. Finally, most of the participants agreed that the DEPLOYMENT FREQUENCY is neither valuable nor impacting their performances (48% and 47% respectively). These results can be visualized in Figure 3.

IV. DISCUSSION AND IMPLICATIONS

The Accelerate metrics are a step forward in evaluating the development process and the performance of development teams. While the Accelerate metrics closely resemble traditional Agile metrics, their combination seems beneficial in industrial settings.

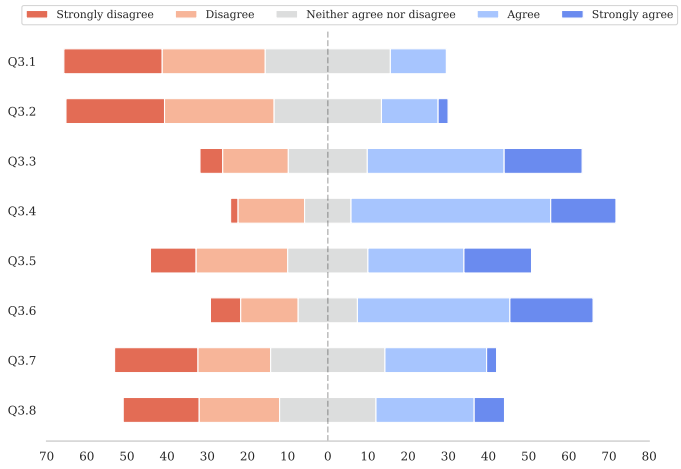

Fig. 3. Perceived Usefulness of the Accelerate metrics (RQ3). On the x-axis is presented the frequency of the answers given in percentage: the percentages of respondents who agree with the statement are shown to the right while the percentage of the ones who disagree is on the left of the 0.

A. Learning From the Accelerate Metrics

Based on our results, we believe that these metrics might be a good proxy to raise alerts for teams experiencing a decrease in performance [11]. For example, the metrics could identify services that require refactoring activities, therefore reducing the technical debt. It is important to note that the metrics are not very relevant when considered individually. However, when combined, they are particularly useful for providing a high-level representation of the performance of the development process. The Accelerate metrics could enable development teams to focus on productivity, particularly on reducing service failures and inactivity.

B. Continuous Improvement

The continuous monitoring of process performance allows developers to identify bottlenecks, detect failures and put in place countermeasures [12]. The information on service failures, and in particular, the frequency of the failures (CHANGE FAIL RATE) and the time needed to recover (MEAN TIME TO RECOVER), slowly changed the developers' mindset. Developers are now more aware of what might happen with non-fully tested code, and they are now more prone to increasing the testing coverage to avoid the occurrence of service failures. The Accelerate metrics make it possible to check "at a glance" if something is malfunctioning, and therefore, it allows for a quicker and more prompt reaction. This can help better address technical debt and improve the overall process performance in the long run.

V. THREATS TO VALIDITY

Our work is subject to different limitations. Therefore, we attempted to mitigate them as much as possible.

To reduce *construct validity* threats, preventing hypothesis guessing and evaluation apprehension [13], in the invitation e-mail, we clearly explained the purpose of the study and asked the participants to answer questions based on their own experience.

To mitigate *conclusion validity* threats, one of the authors coded the responses of the open-ended questions, and another author independently verified the output. Disagreements were discussed and resolved.

To deal with *internal validity* threats, we designed the survey to include only direct questions and, thus, requiring as little interpretation as possible, avoiding misunderstanding that would lead to meaningless answers. Further, the survey has passed through successive validation tasks and a pilot phase to detect any inconsistencies or misunderstandings before distributing the survey on a larger scale.

Regarding the generalization of the results (*external validity*), even if our sample size is significantly larger than that used in prior studies, it only refers to developers working at Vendasta. We reduce this threat by achieving a diversity of participants who answered the survey. However, we cannot make concrete statements about how generalizable the results are for other companies. Another threat that could affect this study is the lack of control over the participants invited to participate in the research. For example, it could happen that only developers interested in the process performances answered the survey. This might bias the results towards a more positive view of the Accelerate metrics.

VI. Conclusion

Our work aimed at assessing the benefits and usefulness of the Accelerate metrics for process performances on teams developing microservices at Vendasta. Only one study investigated the impact of the Accelerate metrics collecting six interviews from one development team. In our study, we surveyed 62 developers from different development teams. Our research questions addressed two aspects: 1) the impact of technical debt on development and delivery, and 2) the participants' perception of the benefits and usefulness of the Accelerate metrics. With this work, we contributed to the body of knowledge by providing the first empirical study on the Accelerate metrics in the context of microservices.

Based on the results of our study, we conclude that the participants considered the Accelerate metrics as valuable, especially to have an overall view of the issues and understand services suffering from technical debt. Although the Accelerate metrics do not directly highlight specific technical debt issues, participants were more prone to reduce their technical debt and test their services better to optimize the value of the metrics. In particular, the metrics optimization led to the decrease of change failure rate and a faster time to restore, resulting in a better quality of service provided to the customers. Moreover, the Accelerate metrics can be beneficial for companies with similar settings. From a practitioner's viewpoint, the results of our work are promising because even with a limited amount of metrics, the company was able to improve the development process and reduce technical debt. We hope that a gamification approach aimed at optimizing the values of the Accelerate metrics might provide an even better result. As an example, developers might try to optimize the metrics by playing a game against each other to reduce the lead time and define the monthly leaderboard of the "fastest developer" or the "cleanest developer" that has the lowest Change Fail Rate.

Future work includes a broader validation of the benefits of the Accelerate metrics, including their actual impact on technical debt, their introduction in tools [14], and their introduction into continuous software maintenance models [15].

Acknowledgment

The authors thank their industrial partner, Vendasta and their employees, for their eager participation in this study, and Pranav Chadramouli, an undergraduate student at the University of Saskatchewan, helped with the figures.

References

[1] C. WH Davis. *Agile metrics in action*. Manning Publications Company, 2015.

[2] E. Kupiainen, M. V Mäntylä, and J. Itkonen. Using metrics in agile and lean software development–a systematic literature review of industrial studies. *Information and software technology*, 62:143–163, 2015.

[3] N. Forsgren, J. Humble, and G. Kim. *Accelerate: The science of lean software and devops: Building and scaling high performing technology organizations*. IT Revolution, 2018.

[4] Florian Auer, Valentina Lenarduzzi, Michael Felderer, and Davide Taibi. From monolithic systems to microservices: An assessment framework. *Information and Software Technology*, 137:106600, 2021.

[5] D. Taibi, V. Lenarduzzi, and C. Pahl. Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation. *IEEE Cloud Computing*, 4(5):22–32, 2017.

[6] D. Taibi and V. Lenarduzzi. On the definition of microservice bad smells. *IEEE Software*, 35(3):56–62, 2018.

[7] V. Lenarduzzi, F. Lomio, N. Saarimäki, and D. Taibi. Does migrating a monolithic system to microservices decrease the technical debt? *Journal of Systems and Software*, 2020.

[8] J. Soldani, D.A. Tamburri, and W. Van Den Heuvel. The pains and gains of microservices: A systematic grey literature review. *Journal of Systems and Software*, 146:215–232, 2018.

[9] V. Lenarduzzi and O. Sievi-Korte. On the negative impact of team independence in microservices software development. In *Int. Conference on Agile Software Development: Companion*, 2018.

[10] M. Sallin, M. Kropp, C. Anslow, J. W Quilty, and A. Meier. Measuring software delivery performance using the four key metrics of devops. In *International Conference on Agile Software Development*, pages 103–119. Springer, Cham, 2021.

[11] V. Lenarduzzi, A. Martini, N. Saarimäki, and D. Tamburri. Technical debt impacting lead-times: An exploratory study. In *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 188–195, 2021.

[12] D. Taibi, V. Lenarduzzi, and C. Pahl. Continuous architecting with microservices and devops: A systematic mapping study. In *International Conference on Cloud Computing and Services Science*, pages 126–151. Springer, 2018.

[13] C. Wohlin, P. Runeson, M· Höst, M. C Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[14] H. Nguyen, F. Lomio, F. Pecorelli, and V. Lenarduzzi. Pandora: Continuous mining software repository and dataset generation. In *IEEE International Conference on Software Analysis, Evolution and Reengineering*, 2022.

[15] V. Lenarduzzi, A.C. Stan, D. Taibi, D. Tosi, and G. Venters. A dynamical quality model to continuously monitor software maintenance. In *11th European Conference on Information Systems Management*, 2017.