Measuring Software Delivery Performance using the Four Key Metrics of DevOps

Marc Sallin [1], Martin Kropp [1], Craig Anslow [2], James W. Quilty [2], and Andreas Meier [3]

[1] University of Applied Sciences and Arts Northwestern Switzerland, Windisch, Switzerland
marc.salin@outlook.com,martin.kropp@fhnw.ch
[2] Victoria University of Wellington, Wellington, New Zealand
craig@ecs.vuw.ac.nz,james.quilty@ecs.vuw.ac.nz
[3] Zurich University of Applied Sciences, Winterthur, Switzerland meea@fhnw.ch

Abstract. The Four Key Metrics of DevOps have become very popular for measuring IT-performance and DevOps adoption. However, the measurement of the four metrics deployment frequency, lead time for change, time to restore service and change failure rate is often done manually and through surveys - with only few data points. In this work we evaluated how the Four Key Metrics can be measured automatically and developed a prototype for the automatic measurement of the Four Key Metrics. We then evaluated if the measurement is valuable for practitioners in a company. The analysis shows that the chosen measurement approach is both suitable and the results valuable for the team with respect to measuring and improving the software delivery performance.

Key words: DevOps, Agile, Metrics, Four Key Metrics, IT-Performance, Case Study

1 Introduction

More and more organizations are adopting DevOps to accelerate delivery speed and improve quality of their software products [1]. The term DevOps first appeared in 2009 in social media coined by Patrick Debois [2]. Bass et al. define the term in their book as "*a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality*" [3]. Companies state that the measurement of DevOps progress is seen as important but also as very difficult [4]. The State of DevOps report, first published in 2014, provides a view into the practices and capabilities that drive high performance in software delivery [5]. The researchers found that only four key metrics (FKM) differentiate between low, medium, high, and elite performers: Lead time for change, deployment frequency, time to restore service, and change failure rate [6]. These four metrics help organizations and teams to determine whether they are improving the overall IT-performance.

As they are strongly correlated with well-known DevOps practices they are also known as DevOps metrics. [5]

The FKM in the State of DevOps report is based on surveys. While a survey approach has the advantage that you can raise highly focused questions, and address a clear target audience, it also comes with several disadvantages: the absence of a clear definition of the measurement, no continuous data, subjective answers, offline data capture and analysis, and extra effort and cost to generate the data. On the other side, using system data provides the advantage that these data are instantly available (e.g. the number of User Stories done in a Sprint), and can be captured and analysed automatically. However, these data may not be complete with respect to the required DevOps aspects (e.g., cultural measures) [7]. This work aims to address these disadvantages by automatically measuring the FKM and evaluate if the automatic measurement of the FKM is of value for practitioners with respect to improving their performance.

We defined two research questions to be answered by this study.

RQ1 How can the FKM be automatically measured?
RQ2 How valuable is the approach to automatically measure the FKM for software development teams?

RQ1 was answered using a multivocal literature review [8] to include both state-of-the-art and -practice literature. To answer RQ2 the findings of RQ1 were operationalized using a prototype. The prototype was used by a development team in an industrial context at Swiss Post and we asked the members of the development team to participate in a survey.

2 The Four Key Metrics (FKM)

This chapter explains the origin of the FKM, describes their original definition and explains why they gained a high popularity in industry. While DevOps has become very popular in industry [1, p. 18] to bring development and operation close together and deploy new software faster into operation, it was unclear how you can measure the improvement in DevOps.

Forsgren et al. searched for a performance measurement of software teams which focus on global outcome in DevOps. That means, in the basic sense of DevOps, firstly, a measurement that does not pit development against operations, by rewarding development for throughput and operations for stability and secondly, focus on outcomes, not output. That means, do not reward people for putting in large amounts of work, but rather measure results that add business value. They found that four key metrics differentiate between low, medium, high, and elite performers [9]. Forsgren et al. defined the metrics as follows:

Deployment Frequency: addresses minimizing the batch size in a project (reducing it is a central element of the Lean paradigm). As this is hard to measure in software, they took the deployment frequency of software to production as a proxy.

Lead Time for Change: defined as "*the time it takes to go from code committed to code successfully running in production*". Shorter time is better because it enables faster feedback and course correction as well as the faster delivery of a fix to a defect.

Time to Restore Service: as failure in rapidly changing complex systems is inevitable the key question for stability is how long it takes to restore service from an incident from the time the incident occurs (e.g., unplanned outage, service impairment)?

Change Failure Rate: the percentage of changes for the application or service which results in degraded service or subsequently required remediation (e.g., lead to service impairment or outage, require a hot fix, a rollback, a fix-forward, or a patch).

In recent years, the FKM have gained large attention and popularity in industry and are applied by many known companies, like Zalando, RedGate, HelloFresh, PBS or Contentful. The publication of Forsgren's book "Accelerate: The Science of Lean Software and DevOps" in 2018 which summarizes their research [10], and the recommendation of ThoughtWorks in 2019 to adopt the FKM in their technology radar [11] has further increased the popularity of the FKM. The DevOps Trends Survey for 2020 carried out by Atlassian shows that nearly half of the respondents leverage the four key metrics [4, p. 24].

3 Multi-vocal Literature Review

Despite interest from industry [4], at the time of writing there is no research which suggests/summarizes how to automatically measure the FKM. To be able to define a broadly accepted definition for the automatic measurement we investigated what other researchers and practitioners did in this area. Usually systematic literature review (SLR) studies are conducted to capture the state of a research topic. However, SLRs focus mainly on research contribution and do not include grey literature from practice (GL) [8]. As a large majority of software practitioners do not publish in academic forums, we also included GL to make sure we get the result of the current state-of-practice in this field. Furthermore, the very current perspective, the relevance for practitioners and the low volume of published research indicates that not only formal literature should be used to cover a topic [8].

The multivocal literature review was conducted according to the guideline of Garousi et al. [8] which considers also the popular SLR guidelines by Kitchenham and Charters [12]. Literature was included if any of the inclusion and none of the exclusion criteria are met (see table 1).

3.1 Systematic Literature Review

The publications of Forsgren et al. (i.e., the book "Accelerate" and the "State of DevOps Reports") are listed in Google Scholar and Research Gate. For the SLR,

Table 1. Multivocal literature review inclusion and exclusion criteria.

Inclusion	Exclusion
Contains more detailed definition than Forsgren et al.	Is not written in English or German.
Contains information about automatic measurement or tooling.	Is not text (e.g., YouTube, or Webinar)
Contains experience report about the automatic measurement	Is a book

the relevant/related research was identified using snowballing starting from their publications. All 93 unique articles which cited the literature about the FKM published by Forsgren et al. were retrieved by following the cited links. Citations from books were not included. 21 articles are not written in English or German and hence were excluded. Only 7 of the 72 remaining articles treated the topic "metrics" and none of them contained more information about the FKM than already presented by Forsgren et al. As no articles from the SLR were included, no data could be extracted and used in the synthesis.

3.2 Gray Literature Review

For the gray literature review (GLR) Google was used as search engine because pilot searches have shown that there is no more narrow scope for the source of information (e.g. only StackOverflow or Medium) which returns results. A pilot search was conducted to find which keywords are used when people are talking about the FKM. This was done by retrieving articles which talk about one of the four metrics (search for "deployment frequency", "lead time for change", "time to restore service" and "change failure rate") and screening the articles to see how the authors bring them into the context of the FKM. As a result, the following search terms were defined to be used for the GLR.

- DevOps metrics
- DevOps metrics accelerate
- DevOps metrics DORA
- four key metrics DevOps
- accelerate metrics definitions

In contrast to the searches within the formal literature, gray literature search returns an exhaustive number of results. Thus, stopping criteria need to be defined [8]. Google has a ranking algorithm which aims to return relevant articles ordered by priority. That means, the most relevant articles are at the top and the following stopping criteria were applied.

- *Theoretical saturation*: As soon five articles in a row did not match the "Is about this topic & contains information" inclusion criteria, the next five articles were screened by only looking at their title. If they were not relevant, the search was ended.

– *Effort bounded*: After reviewing 100 results for a search term, the search was ended.

Initially, 115 articles/search results were retrieved and screened. 43 out of those 115 were not about the topic and 5 were not in text form. 16 unique articles remain which either include a definition or an experience report.

3.3 Results

This section presents the results of the multivocal literature review. The full list of retrieved literature is provided online.[1]

Deployment Frequency: 7/16 articles contain a definition for deployment frequency. As this metric is already well defined by Forsgren et al. as *deployment of software to production*, the definitions do not widely diverge. They have in common that "number of deployments/releases in a certain period" are counted. Some state that they only count successful deployment (but successful is not defined) and some explicit mention that they count deployments to production. For the purposes of automated measurement, a deployment is defined as a new release [2] As this is a speed metric, every deployment attempt is counted as deployment even if it was not successful.

Lead Time for Change: 9/16 articles contain a definition for lead time for change. Like the deployment frequency, the definition of Forsgren et al. does not leave much room for interpretation although some deliberately took approaches diverging from that of Forsgren et al. All suggestions based on the original FKM definition measure the time a commit takes until it reaches production, the only difference is how they aggregate (i.e., mean, median, p90 etc.). Today it is default practice to use a version control system for source code. To make an adjustment to the software system a developer has to alter source code and to put it under version control. Hence, the commit[3] is defined as the "change". Thus, the lead time is given by the time span between the timestamp of the commit and the timestamp of the deployment, as defined in in 3.3.

Time to Restore Service: 8/16 articles contain a definition for time to restore service. Five of them define the time to restore service as mean time for closing an incident in a certain period. One suggests using chaos engineering (i.e., introduce a failure and measure how long it takes until it gets discovered and resolved), there is a suggestion to periodically poll the "status" and record how long it takes when the status indicates degradation until the degradation gets restored (but do not mention from where the status is taken). The last suggestion made by two articles assumes that the time to restore service should be calculated for failed releases and thus suggests identifying "fix releases" and measuring how long it takes from one release to the following "fix release". The

[1] `https://1drv.ms/x/s!ApmGN3k-vuHI1ZxB8z9SnoO0r0t_vw?e=qAuxgW`

[2] A delivered version of an application which may include all or part of an application. [13, p.296]

[3] Depending on the used version control system this is called e.g. "commit" or "check-in".

reasons for a failure are manifold, and frequently rely on human interpretation of what constitutes "failure" and "fix", which makes it difficult to fully automate this metric. Provided that a team has an incident management, the calculation via incidents is an interesting approach. Since the incident creation could also be automated, this approach allows a mixture of manual and automated failure recognition. For this work, we define the time to restore as the time between incident creation to closing the incident, like this is stated by the majority of articles found. This choice was made because there is already an incident management in place, which can be used to gather the data and this seems so far to be the most reliable source of data.

Change Failure Rate: 9/16 articles contain a definition for change failure rate. The different suggestions are listed below.

- Percentage of releases that were followed by a "fix release".
- Count of hot fixes in commit messages.
- Detect failures by using monitoring metrics and divided by deployments.
- Manually mark a deployment as successful or not.
- Count rollbacks divided by deployments.

To measure change failure rate, first, it has to be defined what a change is. In all identified articles a change is indicated by a deployment. Accordingly, the change failure rate is the ratio of change failures to deployments (see 3.3). The next challenge is to identify a failure and attribute it to a change. Unlike for the time to restore service, the incident management cannot be used for failure detection as, according to Forsgren et al., a change failure includes all cases where subsequent remediation was required.[4] Especially for development teams with a good software delivery performance, the team itself will be responsible for the deployment and any resulting failures will be fixed immediately without an incident ever existing. As we assume a low change failure rate in the context of our case study of Swiss Post, we decided to use for our measurements the manual classification of a deployment as a failure by the development team.

Summary: The velocity metrics are more precisely defined and thus the automatic measurement is easier and more straightforward to derive. This is also reflected in the articles found. With the toolchain used by the development team, the measurement of the speed metrics can be completely automated. The stability metrics are less well defined, and unlike the velocity metrics, the boundaries can be drawn less precisely. The literature provided various approaches, but the approaches that would have allowed a fully automated measurement do not capture all relevant aspects of the metrics. For this reason, we have chosen to use only partial automation for measuring the stability metrics. We assume that the change failures are less manifold than failures in general and thus suggest the creation of a taxonomy of change failures, which will be the enabler for tools and concepts to automatically detect them.

[4] This could be, for example, an automated rollback which is never visible in the incident management system

4 Measure the Four Key Metrics

Based on the earlier definitions section 3 an application was built to measure the FKM automatically. The prototype application is divided into two main parts (green) and several data sources (blue). One is the collector, and the other is the aggregator. The collector is responsible to gather the necessary raw data, transform them and write them in a not compressed manner to the storage of the aggregator (i.e., do no calculations like average). The aggregator enables different calculations and visualizations. This separation aims to enable the usage of different collectors (e.g., for applications which are built and deployment with another toolchain) and to keep the flexibility of having different ways of calculating the metrics (e.g., use the median instead of the mean or use other time periods). The Fig. 4 shows the components of the prototype.

The collector part was implemented using Jenkins[5] (Host/Execute in regular intervals) and PowerShell[6] (collection logic). The aggregator part was Splunk[7] (use an index as storage, do calculations using the Splunk Query Language and visualization with a dashboard). The resulting UI is shown in Fig. 2.

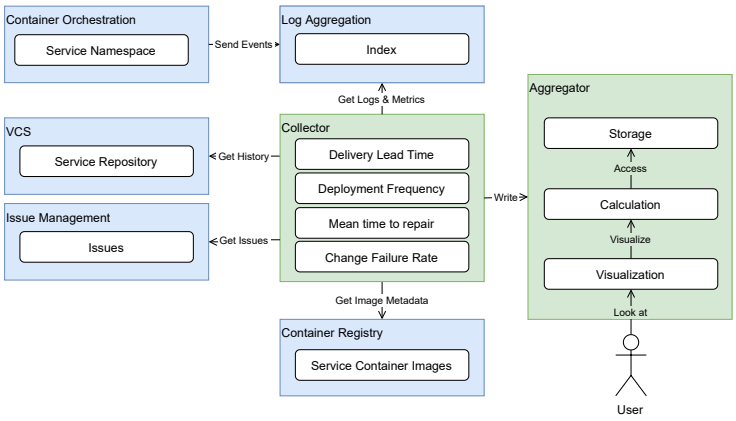

Fig. 1. Concept of prototype to measure the FKM.

Deployment Frequency: the deployments are detected by using the pulled event from OpenShift, which is logged if a Docker image is pulled from a registry. Those events are sent to Splunk from where the collector gets the data.

Lead Time for Change: the deployments are collected like described in "Deployment Frequency". The logged event contains the name of the Docker image and the version. The comm it hash from which the Docker image was built is retrievable from Artifactory by using the Docker image name and version. The source code repository URL is also attached as metadata to the Docker image.

[5] `https://www.jenkins.io` (25.04.2021)
[6] `https://github.com/PowerShell/PowerShell` (25.04.2021)
[7] `https://www.splunk.com/` (25.04.2021)

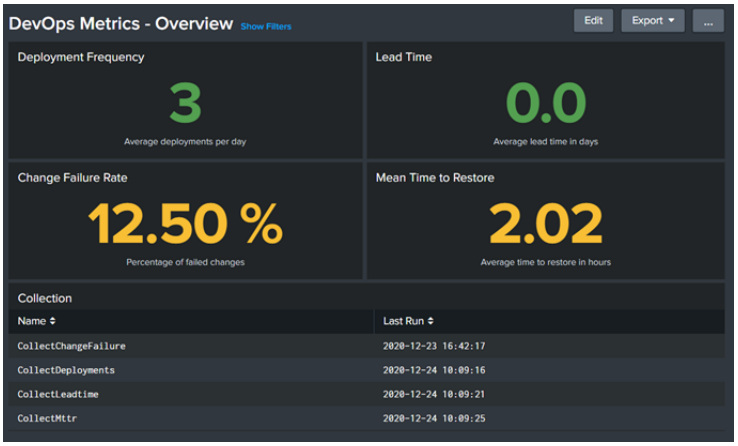

Fig. 2. Dashboard showing the automatic measured FKM.

With the commit hash of the current deployment and the last deployment, as well as the repository URL, all commits which were newly deployed to production are retrieved and the lead time for each commit is calculated.

Time to Restore Service: the Scrum team uses Jira for issue management. Jira issues of type "incident" are retrieved and the time to restore service is calculated by taking the time passed from creation of the incident until it was resolved.

Change Failure Rate: how to collect the deployments was described in "Deployment Frequency". Additionally, failed changes are registered by the team, using a web-based form (implemented with Jenkins). With those two ingredients, the percentage of failed deployments is calculated.

5 Case Study

To investigate the RQ2 "How valuable is the approach to automatically measure the FKM for software development teams?", we conducted a survey after the development team had worked with the prototype for three weeks. For this, the prototype was presented and explained to the team. They decided to look at the metrics once a week as part of an already existing meeting and discuss them.

5.1 Case Description

The study was conducted at the Informatics/Technology function unit of Swiss Post Ltd. Swiss Post Ltd is the national postal service of Switzerland. The group consists of strategic subsidiaries which operate in the core markets of Swiss Post Ltd and function units which are affiliated to Swiss Post Ltd. [14].

In 2019 the Informatics/Technology function unit had around 1200 full-time equivalent employees with about 330 software developers. The unit runs only

projects for internal customers i.e., the IT department does not offer services for customers outside of the group. The Scrum team consisting of 10 people looked at in this study is located in the Informatics/Technology function unit and works for Logistics Services in the area of parcel sorting. Logistics Services is one of the subsidiaries of Swiss Post Ltd and is among other things responsible for parcel sorting and delivery. In 2020 Logistics Services delivered 182,7 million parcels [15]. The Informatics/Technology function unit is under pressure, that new products should be delivered earlier to the customer, and IT must be able to react faster to changes in the environment. At the same time, the customer expects consistently high quality. To achieve these goals Informatics/Technology function unit is undergoing an agile transformation. Beside adopting agile methodologies like Scrum and Kanban, DevOps practices are being introduced as part of this agile transformation. The Informatics/Technology function unit is mainly organized project driven but the Scrum team which participated in this study is a stable product team. In 2020 they started to work on a new software system, which is one of the core systems for the sorting of parcels. The team also started to invest in their tool-chain (Figure 5.1) and began adopting certain DevOps practices. However, so far, no actions have been taken by the management and the team to measure the progress of the transformation process and the DevOps practices with respect to its improvements; so the team is also not able to track the progress of improvements.

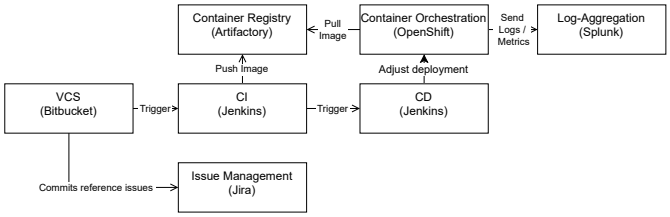

Fig. 3. The toolchain used by the Swiss Post Scrum team.

5.2 Methodology

The research question was divided into sub questions to be answered by the survey. In order to be valuable, a metric should be valid, this concern was addressed by the sub questions one and two. A metric is considered valuable if the team can act on it i.e. it leads to effects. This was addressed by sub question three. The sub question four aimed to capture the subjective perspective of the team onto the value of the automatically measured FKM.

1. Are the FKM a valid way of measuring the speed and stability for software delivery?
2. Can the FKM be used as an indicator for advancing DevOps adoption?
3. What is the effect of measuring the FKM on the teams?

4. Does the development team find value in measuring the FKM?

The survey consisted of four parts: demographics, metric measurement, effect of metric measurement and usefulness of metrics. In the metric measurement part, the participants were asked for their opinion about the FKM and if they think that the metrics as defined by Forsgren et al. measure what they claim to. Furthermore, they were asked how good the automatic measurement implementation reflects what the metrics should measure. The effect of the metric measurement part asked what effects, if any, they expect if the metrics would be measured long term.[8] Finally, the participants were asked how likely they would recommend another team to use the automatic metrics measurement and what metric they consider the most important to be automatically measured.[9]

The questions and our analysis is principally based on Likert scales, and is therefore a quantitative approach based on self-reported experience and perception. After each Likert scale question, there was the possibility to optionally explain the rating in free text.

Six out of ten team members participated in the survey. Among them are four developers and two software architects, aged between 25 and 44. Four of the participants stated that they already knew about the FKM before they were introduced by us.

5.3 Results

This section presents the results of the survey about the automatic metrics measurement. The results are provided online.[10]

Metrics Measurement Fig. 4 shows what the participants think about the metrics defined by Forsgren et al. In general, the participants agree (statement 1: 2× strongly agree, 4× agree) that the FKM are a valid way of measuring the speed and stability for software delivery. The two speed metrics are generally seen as a valid way of measuring the software delivery performance (statement 2: 2× strongly agree, 3× agree, 1× disagree). There is slightly less agreement about the stability metrics (statement 3: 1× strongly agree, 3× agree, 2× somewhat agree). The FKM were explicitly picked to make sure teams do not trade-off quality for speed. However, it seems there is a piece missing as the participants are sceptical about the stability metrics showing when a team does this trade-off (statement 4: 1× strongly disagree, 1× somewhat agree, 2× agree, 1× strongly agree).

For the implementation of the automatic measurement there was broad agreement that the measurement of the speed metrics is sufficient and moderate agreement about the stability metrics (see Figure 5).

[8] The measurement period was to short to ask for effects that have already occurred.

[9] The full survey can be retrieved here `https://1drv.ms/b/s!ApmGN3k-vuHI1ZVZuj-pZY76IvhC_Q?e=AaxwMe`

[10] `https://1drv.ms/x/s!ApmGN3k-vuHI1ZtdjwlKBlPYInsY2g?e=V2aBMD`

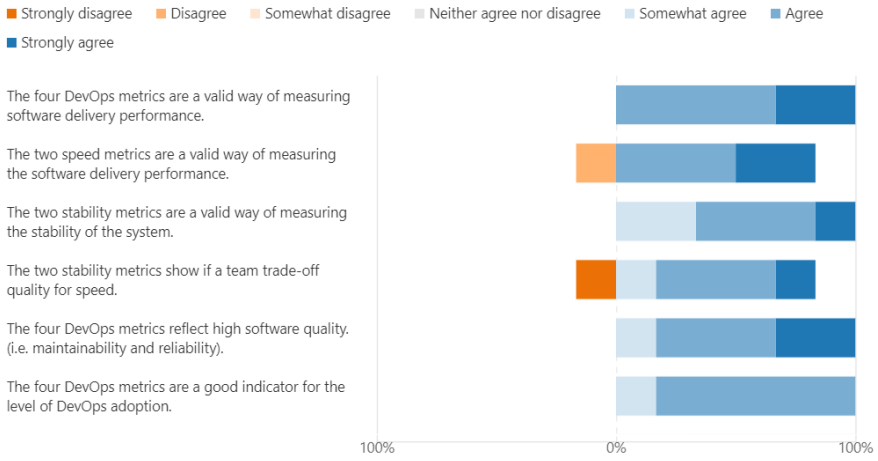

Fig. 4. Participants agreement about the validity of the FKM, as originally defined by Forsgren et al.

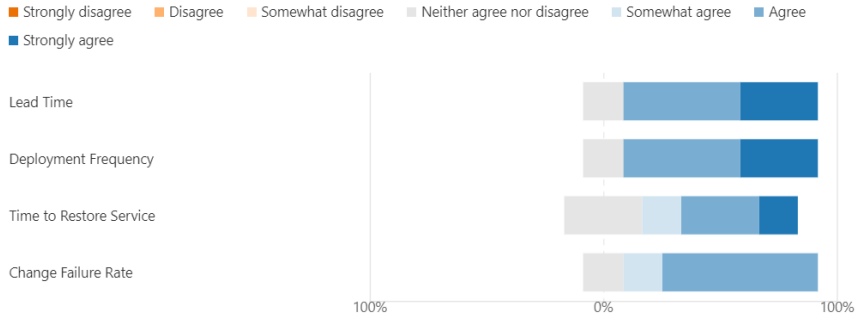

Fig. 5. Participants agreement about whether the implemented automation is able to capture the correct measurement.

Effect of metric measurement Figure 6 the answers to the expected effects are shown. The left part shows the answers to the question whether the participant expect an effect on the team and/or on the personal behavior. The left side shows what kind of effect they expect.

The described effects expected for the personal behavior were: Pay more attention to get the pull requests merged quickly, greater use of feature flags, better root cause analysis, more team spirit, feeling more responsible for the work, source of motivation to improve. The effects effects expected for the team behavior were: More attention for the software delivery process, less deployment pain and anxiety, more relaxed environment, encourage some technical practices, putting some pressure to some team members, source of motivation to improve.

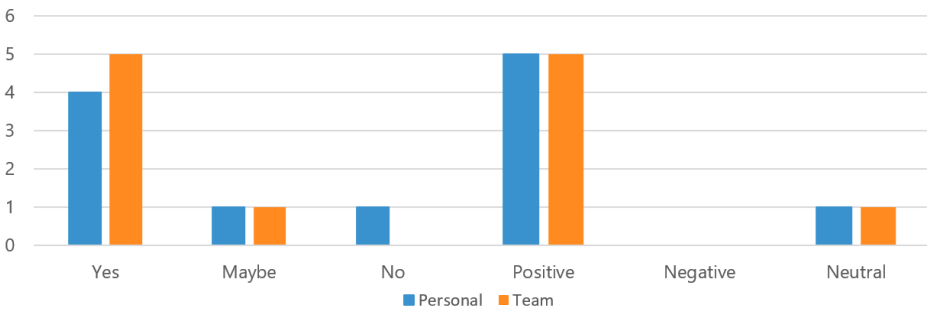

Fig. 6. Expected effects of long term measurement.

Usefulness of the Metrics The question "On a scale from 1-10, how likely are you to recommend the metrics measurement to another team?" was answered with an average score of 8.3 (max: 10, min: 6, sd: 1.5). The participants ranked the Lead Time for Change as the most important metric to be automatically measured, followed by Change Failure Rate, Time to Restore Service and the Deployment Frequency on the fourth rank. The participant with the value of six explained this score in the free text answer. He wrote that he thinks the most teams do not yet have the mindset nor the tools to start measuring.

6 Discussion

6.1 RQ1: How to automatically measure the FKM?

This question was addressed with a multivocal literature review. We identified no scientific literature which investigated the automatic measurement of the FKM. The gray literature review revealed sixteen articles which described aspects of the automatic measurement. Nine of them were published in 2020, three in 2019 and one in 2018, which we explain with the growing interest in DevOps.

The definitions by Forsgren et al. for the stability metrics (change failure rate and time to restore service) are not as clear as the definitions of the speed metrics (deployment frequency and lead time for change) and therefore the suggestions for how to perform the measurement are more diverse for the stability metrics than for the speed metrics. How to automatically measure the speed metrics only differs in detail but still, they are context sensitive i.e., some practices applied by a team can have influence on the definition used and also the measurement implementation (e.g., feature flags, canary releases or Blue/Green deployments).

Worth to mention is the fact that automatic measurement is only possible if the processes are automated. That means only teams which already invest in their automation will be able to use the automatic measurement. Thus, a team might start with tracking their FKM with a survey to get a baseline [7] and as they advance, they switch to an automatic measurement.

6.2 RQ2: How valuable is the approach to automatically measure the FKM for software development teams?

We addressed this question with a prototype and a survey. The sub questions to this research question are discussed in the following sections.

Are the FKM a valid way of measuring the speed and stability for software delivery? In general the participants agreed that the metrics defined by Forsgren et al. are valid. There was slightly more agreement about the speed than the stability metrics. This might be the case because the speed metrics are easier to measure and more clearly defined. Participants are skeptical about the stability metrics showing when a team trade-off quality against speed. A possible reason is that the metrics will show this trade-off not in short but in long term. If a team decides to not constantly refactor the code base, in the short term they will be faster and not lose stability. However, in the mid to long term the technical dept will lead to lower pace and to lower stability [16]. Another reason is that change failure rate and time to restore service do not capture all quality aspects. Lets imagine a team skips all regression test non-critical parts of the system. They get faster but more bugs are discovered in production, which makes the perceived quality of users lower. But the stability metrics will not show this if the bugs do not lead to failures. Although the stability metrics are valid to capture the stability of a system, each system has other quality attributes which should not be traded-off for speed. As they are individual, each team might define additional metrics to capture these attributes.

Can the FKM be used as an indicator for advancing DevOps adoption? This question was addressed with a statement with a Likert scale rating. The participants generally agreed but not strongly. The reason gets unveiled when looking at the free text answers. There is general agreement that the FKM are good representatives of DevOps adoption but do not cover every aspect. One aspect mentioned is the provisioning of infrastructure. A DevOps team might be fast with existing services but slow when creating a new one. That might be fine if it's a very rare event, otherwise it should be optimized as well. In his article, Cochran talks about his research about "developer feedback loops" and mention metrics which might not be covered by the FKM like "Find root cause for defect", "Become productive on new team", "Launch a new service in production" [17]. However, there is a need to consider that Forsgren et al. do not postulate that the FKM cover all aspects but say that they are sufficient to divide the teams into clusters. Thus, it would have to be examined whether the high/elite performers correlate with efficient developer feedback loops.

What is the effect of measuring the FKM on the teams? The participants mostly agreed that they see value in automatically measuring the FKM and expect that the long-term measurement will lead to positive effects on the personal and on the team behavior. But they also state that the FKM do not cover all aspects which are considered as important for quality, speed, and DevOps in

general (see also the sections above). It was mentioned that the intrinsic motivation of getting better will let the team take the necessary actions to improve the metrics. The intrinsic motivation is important because one should be well aware of Goodhart's law which states that "When a measure becomes a target, it ceases to be a good measure." [18]. It is not recommended to give a development team the goal of improving metrics and rewarding them with external incentives. Furthermore, a team might need to be guided to improve their FKM. The participants who have a noticeable amount of experience in software engineering, agile and some experience with DevOps only moderately agree that they are able to influence the FKM (2× somewhat agree, 3× agree, 1× strongly agree) and that they know which practices and techniques correlate with each of the DevOps metrics (3× somewhat agree, 2× agree, 1× strongly agree).

Does the development team find value in measuring the FKM? From the stated expected positive effects and the high recommendation score the conclusion follows, that the team sees value in measuring the FKM. However, this have to be seen in the context of this team. One of the participants states that he belief that a team has to have already the "right" mindset to get a value from the measurement. If a development team which want to improve it's software delivery performance and/or DevOps adoption the automatic measurement of the FKM are a valuable tool.

The metric with ranked with the highest priority to be measured was the lead time for change. That one with the next was change failure rate. Which indicates that there is the motivation to see how fast one can deliver, however, stability should not suffer as a result.

6.3 Limitations

We are aware that the study has several limitations. The SLR part of the MLR returned no results i.e, no scientific literature but only gray literature was included. The lack of scientific literature is maybe due to the applied methodology (i.e., there are articles, but they are not listed to cite the origin articles used for snowballing). The definitions found are presented in gray literature, those articles are usually less reliable as they do not apply scientific methods, not provide many details and are representing the subjective view of the author. Furthermore, it is possible that important gray literature was not included because a GLR is not exhaustive but certain the stopping criteria are applied.

The research was carried out as a case study. The the sample size was small and homogeneous. Participants of the survey were already high performers according to FKM and had already invested into their CI/CD processes, else the automatic measurement wouldn't have been possible. Hence, the results of the study are not representative and generalization is only possible to a limited extent. Furthermore, the study is highly context specific to the Swiss Post environment, which also limits the generalization. But might still be helpful to companies with similar setups. Due to time constraints, the duration in which the team made use of the metrics was too short to ask about perceived effects

and we asked about expected effects. It has to be considered that those effect might not show up as expected.

6.4 Summary

The findings indicate that the suggested automatic measurement of the FKM is a good starting point and a valuable tool for intrinsically motivated software development teams, which want to improve software delivery performance and show their DevOps adoption. But the FKM do not cover every aspect e.g. the developer feedback loops are not covered. Hence, it is important that the development team does not only focus on improving the measurements.

6.5 Outlook

The study found that it is possible to meaningful measure the FKM automatically and the software developers team see it as valuable, the prototype going to be rolled out for all development teams at Swiss Post (i.e., all teams which create containerized applications will be able to use it). Professionals and researchers outside of Swiss Post might adapt the suggested definitions and ways to automatically measure the FKM, to build tools for measuring the FKM in their context.

7 Conclusions

In this work we investigated how the four key metrics defined by Forsgren et al. can be measured automatically and if practitioners think that the metrics are valuable. We conducted a multivocal literature review to reveal how the four key metrics are measured by different companies in the industry and in other research projects. Based on those findings, we created a prototype to automatically measure the metrics. The prototype was used by a Scrum team at Swiss Post for three weeks. Afterwards, we collected their experience by using a survey. The participants of the survey stated that they think that the FKM are a valid measurement for software delivery performance and that they would recommend the automatic measurement for another team. However, they also stated that the FKM are not capturing every important aspect e.g. how fast can infrastructure be provisioned. Despite of the maturity of the team in terms of experience with Agile, DevOps, software engineering and their ranking according to the FKM, it was discovered that the participants only moderately agreed that they think they are able to influence the metrics and that they know what practices to apply to improve. This finding suggests that especially less mature team need guidance to be able to improve on the FKM as they are a lagging measurement and do not directly suggest any actions.

References

1. StateOfAgile: 14th annual STATE OF AGILE REPORT. Technical Report 14 (2020)
2. Mezak, S.: The Origins of DevOps: What's in a Name? (2018)
3. Bass, L., Weber, I., Zhu, L.: DevOps: A Software Architect's Perspective. 1st edn. Addison-Wesley Professional (2015)
4. Atlassian: 2020 DevOps Trends Survey. Technical report, Atlassian (2020)
5. PuppetLabs: 2014 State of DevOps Report. Technical report, PuppetLabs (2014)
6. PuppetLabs: 2019 State of DevOps Report. Technical report, PuppetLabs (2019)
7. Forsgren, N., Kersten, M.: DevOps Metrics. Queue **15**(6) (12 2017) 19–34
8. Garousi, V., Felderer, M., Mäntylä, M.V.: Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. arXiv (2017)
9. Forsgren, N., Smith, D., Humble, J., Frazelle, J.: Accelerate: State of DevOps. Technical report, DORA (2019)
10. Forsgren, N., Humble, J., Kim, G.: Accelerate: The Science of Lean Software and DevOps. 1st edn. IT Revolution Press (2018)
11. ThoughtWorks: ThoughtWorks Technology Radar - Four Key Metrics (2019)
12. Kitchenham, B.A., Charters, S.: Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report (2007)
13. ISO/IEC, IEEE: ISO/IEC/IEEE 24765:2010 - Systems and software engineering – Vocabulary. Iso/Iec Ieee **2010** (2010) 410
14. : Group Struture - Swiss Post (2020)
15. Swiss Post: Post verzeichnet Allzeitrekord von 182,7 Millionen Paketen (2021)
16. Behutiye, W.N., Rodríguez, P., Oivo, M., Tosun, A.: Analyzing the concept of technical debt in the context of agile software development: A systematic literature review. Information and Software Technology **82** (2017) 139–158
17. Cochran, T.: Maximizing Developer Effectiveness (2021)
18. Strathern, M.: 'Improving ratings': audit in the British University system. European Review **5**(3) (7 1997) 305–321