INSTITUTO
UNIVERSITÁRIO
DE LISBOA

KPI’s for Evaluation of DevOps Teams

Marta Alexandra Castro Gomes

Masters in, Computer Science and Business Management

Supervisor(a):
PhD Rúben Filipe de Sousa Pereira, Assistant Professor,
ISCTE – IUL

Joint Supervisor(a):
MSc Miguel Ângelo Rodrigues da Silva, Invited Senior Assistant,
ISCTE – IUL

October, 2022

ACKNOWLEDGEMENTS

Firstly, I must thank to my father, Jorge, and my mother, Helena, for believing in my potential and for providing me with a good education and the means to be the successful person that I am now. To my brothers, Jorge and Guilherme, thank you for looking at me as a role model, it makes me want to give a good example to you guys.

To my boyfriend, Francisco, I appreciate all the support and understanding while I was working on this dissertation.

Thanks to my friends, Mário and Diogo, for the company and support for those first years of studding. To João and Paulo, thanks for the support in the last years of university. To Mariana, a big appreciation for all the hours to spent reading this work.

Finally, I must give a big thanks to my advisor, Professor Rúben Pereira, for being always available to questions, revisions, and debates. Also thank you for the pressure, sometimes it is needed. To my second supervisor, Professor Miguel Silva, I really appreciate for all the ideas for this dissertation, the opportunity you gave me and for seeing potential in me.

Without all of you, this could not have been possible.

RESUMO

Numa era de transformação digital, onde nas últimas décadas tem aumentado a necessidade de rapidamente ser abraçada a mudança, é importante que as empresas façam a sua transformação para que continuem a ser competitivas. Para isto, a organização tem adotado metodologias ágeis que permitem às empresas ganhar vantagem no processo de atingir os seus objetivos. Uma das metodologias ágeis mais exploradas recentemente é DevOps, desta forma, medir a performance das equipas de DevOps é uma necessidade cada vez maior para as empresas. Com este ponto, surge a necessidade de as empresas terem um conjunto de métricas que podem usar para medir a performance das equipas. Para isto, é feita uma pesquisa para perceber que métricas estão já identificadas em estudos existentes. Depois disto, é feito um caso de estudo para perceber que pontos deve uma equipa melhorar.

Palavras-Chave: DevOps, equipas, performance, medir, métrica, indicadores chave de performance, indicador

ABSTRACT

In an era of digital transformation, the need to quickly embrace change in order to become more competitive has increased over the last few decades. So, organizations have adopted agile cultures that allow companies to gain a leverage when reaching their goals. One of the most recently explored agile cultures is DevOps, this way, measure DevOps teams' performance is a bigger and bigger need of the organizations. With this, it is important that the companies have a set of KPIs that they can use to measure teams' performance. So, this research is done, to find the KPIs that are identified in the studies that already exist. Then, a Case Study is put in practice, to determine which points should a team improve on.

Keywords: DevOps, teams, performance, measure, metrics, kpi's, indicator

PUBLICATIONS

In this section, is presented the papers submitted and accepted in the context of this research.

Table 0-1 - Submitted Papers

Paper	Published in	Index	Decision
KPI's for evaluation of DevOps teams	Information Systems and Technologies. Lecture Notes in Networks and Systems	Q4	Accepted

TABLE INDEX

FIGURE INDEX

CONTENTS INDEX

LIST OF ABREVIATIONS AND ACRONYMS

BA – Business Analyst

CS – Case Study

DB – Database

IT – Information Technology

ITV – Iteration Team Velocity

KPI – Key Performance Indicator

PO – Product Owner

RTV – Release Team Velocity

SLR – Systematic Literature Review

USA – United States of America

UX/UI – User Experience/ User Interface

1 INTRODUCTION

Currently, the demand for speed and agility is growing every day, improving the competition [1–4]. Thus, organizations have endorsed agile cultures with the intention of increase the velocity of their response to the eminent change, making it easier for the collaboration in the organization [5–7] and secure the innovation [8]. This way, the agility can give organizations the skill to “swiftly and efficiently respond to internal and environmental organization changes” [4], this is where DevOps became a critical way of working for companies [9]. Today, it is important not only create a variety of features but to do it with high quality [10].

Several agility cultures exist but diverge from each other [11]. One of the most recently explored and adopted [11] is DevOps. DevOps changes the way people work [12], making easier the communication between the team members [13–15]. Thus, is important to be conscious of the various practices that can be adopted (in the culture, automation, measurement, and sharing scope [1]) and the factors that can ease the culture adoption [3] then, measure the performance to understand which ones give companies the best performance.

Human performance “makes modern business-critical systems robust and resilient” [16], being this one of the most important things to ponder, by DevOps teams, in the process of software development [17]. A conceptual map is presented in Figure 1-1 to better explain the context of DevOps team performance.

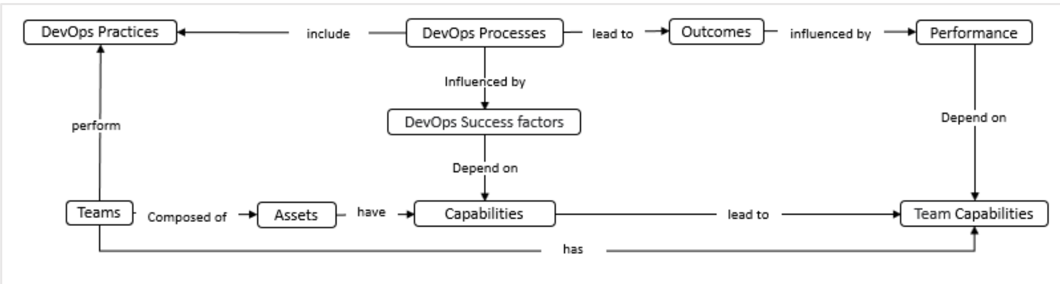

Figure 1-1 - Conceptual Map for Performance of DevOps teams

Therefore, this research aims to understand which are the existing Key Performance Indicators (KPI’s) to measure DevOps teams’ performance and apply them to a DevOps team in an Information Technology (IT) area through a period and with that information determine which factors lead to possible changes in the team’s performance. Initially a Systematic Literature Review (SLR) was performed. The SLR is an appropriated methodology to synthesize what peers have been proposed in previous investigations. This SLR brings the research works, experiences and studies to answer a specific question and topic [18]. On a second phase, after the SLR and after having all the KPI’s, a Case Study (CS) is going to be conducted to find factors that can explain possible variations that can be found in the results of the different KPI’s. A CS “enables the researcher to examine the data within a specific context” [19].

2 STATE OF THE ART

2.1 BACKGROUND

2.1.1 DEVOPS

The authors V. Zamfir, M. Carabas, C. Carabas, N. Tapus [9], present in their article a DevOps definition as a culture "that advocates communication, collaboration and integration between development and operations teams to solve critical issues, such as fear of change and risky development" [9]. With this in mind, over the past few years, companies have been changing to paradigms with faster capability of answering the needs of the market and gain productivity [20–22], being the main goal, the acquirement of the higher level of integration and automation [15,23].

A gap that DevOps came to fulfil, are the 4Cs, communications, co-operation, culture, and collaboration [24,25]. DevOps includes agile principles and practices, and those must be used whenever is possible [26,27].

2.1.2 DEVOPS TEAMS PERFORMANCE

DevOps teams became popular in mid 2010 [28] and is a joined operations and development teams and environments [5,29], being multidisciplinary teams [30,31] that can bridge the multidisciplinary problems that may exist [32]. The point is to have in every DevOps project, both development and operational teams aligned with their objectives [33].

The development part of the team is responsible for the implementation of the technical part of the project (as fast as possible), resolve problems with production and transfer that knowledge to the operational team [9,33]. Operational teams usually are responsible for support [13] keeping the systems available and stable [9].

The roles of the DevOps teams usually include a steering committee, scrum master, product owner, developers, research team, User Experience/ User Interface (UX/UI), design team and a DevOps and infrastructure analyst [34]. The members of this team can work directly with the customer, decreasing the time and failure [35].

DevOps teams should consider some specific processes that include some practices since those are influenced by the DevOps success factors [36–38]. DevOps teams have different capabilities, depending each one of them on the personal skills of every member and the performance of those teams must consider these same capabilities.

2.1.3 KPI'S

KPI's have the objective of measure and monitor the process of software development. In some cases, depending on the factors on different companies, KPI's can be used to compare performances [39]. The use of KPI's to teams, has not been studied yet and so, the use of them by the organizations is far from clear [24,39,40].

This said, the adoption of agile culture can vary, interventions and strategies can change, this way, some KPI's work better than others and the KPI's take in consideration the team, its projects, the environments, and their agile adoption. No KPI to measure performance can be taken on their own.

2.2 RELATED WORK

Before proceeding with the investigation is important to assess and synthesize existing findings in the field. Therefore, a SLR was performed to collect the most relevant studies. To increase the scientific rigor, we have followed the SLR guidelines proposed by Kitchenham [18]. Table 2-1 details the steps performed.

Table 2-1: SLR Methodology

Outlining SLR	Conducting SLR	Reporting the Review
➢ Organizations have adopted agile with the intention of increasing the velocity of response to change. ➢ Performance "makes modern business-critical systems robust and resilient" [16]	**Initial Filter** ➢ 148 Articles **Availability Filter** ➢ 97 Articles **Manual Filter** ➢ 86 Articles	**Report the findings.** ➢ Will be identified the practices that must be followed by DevOps teams. ➢ Point the factors that can enable and prevent teams from adopting agile culture. ➢ Setup the benefits of using agile culture. ➢ Identify and describe the KPI's found to measure performance of DevOps teams. ➢ Analyse a real team, apply them the KPI's found and try to determine factors that influence the team's performance
Objective of the review ➢ Explore what has been investigating in the scope of DevOps team performance	**Perform Data Extraction and analysis of the sample.** ➢ Per Year ➢ Per Country ➢ Per Type of Article	
Review protocol ➢ Search String and Filters, repositories, applying the inclusion and exclusion criteria.		

2.2.1 PLANNING THE REVIEW

The review protocol begins with literature research using a search string into a set of chosen libraries to get the maximum number of studies that could address the proposed research. The libraries and search strings used are listed below.

Libraries: Scopus, ACM Digital Library, IEEE Xplore, EBSCO, and Web of Science.

Search Strings: devops AND (teams OR team) AND (dashboards OR dashboard) AND (measure OR measurement OR metrics OR metric OR KPI's OR indicator)

After that, exclusion criteria must be applied to filter the obtained documents. Our filtering criteria are presented in Table 2-2.

Table 2-2: Filters and exclusion criteria

Filter Name	Criteria
No Filter (F0)	Original search of selected keyword on Full Text
Filter One (F1)	Negate the keywords healthcare, blockchain and cloud
Duplicates	Search for articles found in duplicate (same article found in more than one Database (DB))
Availability Filter	Exclude all articles that could not be downloaded
Manual Filter	Filter manually to retire articles out of the scope and include others found out of the DB's.

After the application of this criteria, the first set of studies is obtained. At the end, these studies are read to obtain the final set to perform this review. This Section corresponds to the second step of the SLR methodology. It has been applied the review protocol previously defined and performed an analysis of the studies found.

2.2.2 CONDUTING THE REVIEW

This Section corresponds to the second step of the SLR methodology. It has been applied the review protocol previously defined in Section 3 and performed an analysis of the extracted studies.

The application of the needed search string in the listed libraries, with the filtering exclusion criteria is presented in Table 2-3.

Table 2-3: Filter application on studies found

DB	F0	F1	Duplicates	Availability Filter	Manual Filter
IEEE Xplore	147	64	64	-	-
ACM Digital Library	100	33	28	-	-
Scopus	15	9	8	-	-
EBSCO	517	56	47	-	-
Web of Science	2	2	1	-	-
TOTAL	781	164	148	97	86

At the end of the DB search and duplicates analyse; 148 articles were obtained. Since some of them could not be find or download, a total of 97 studies were analysed. The final step was eliminating the ones that were out of the scope and add the ones that were firstly analyse before the start of the research, with this, a final of 86 articles compose this assignment.

2.2.3 DATA EXTRACTION ANALYSIS

Looking at Figure 2-1 it is possible to see the distribution over the years of the studies which deal with the keywords. It is a recent topic, having most of them being done since 2019 to 2021.

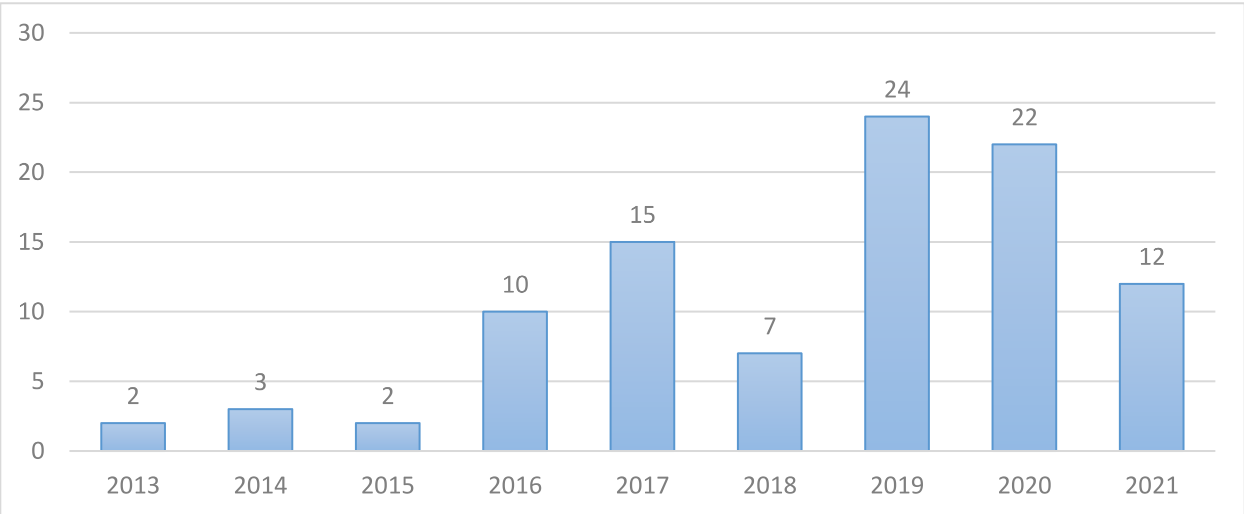

Figure 2-1 - Distribution of articles per year

Next, in Figure 2-2 it is possible to see the distribution of the articles per country, this way it is possible to see that the Unites States of America (USA) is the country that has more articles, being followed by Spain.

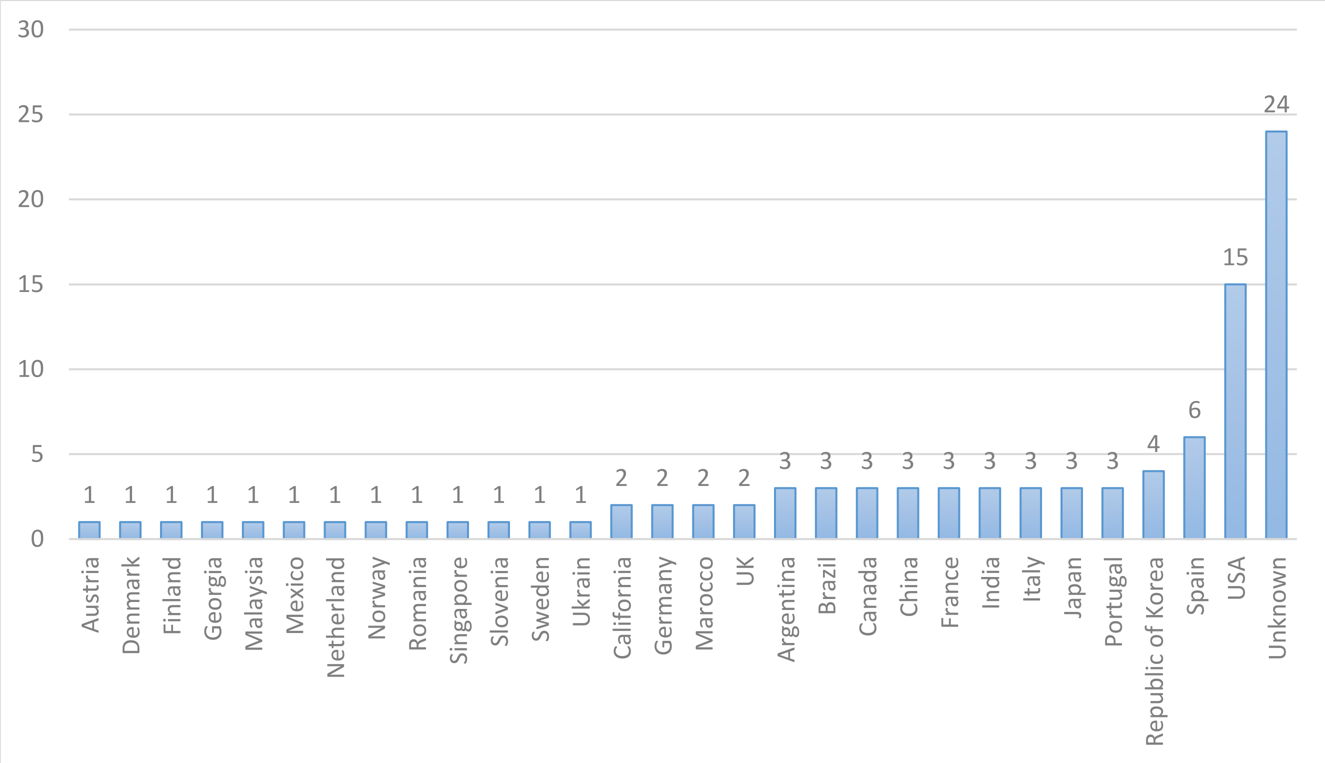

Figure 2-2 - Distribution of articles per country

About the distribution of articles found per type of article, in Figure 2-3, its visible that in comparison to the other type of articles, the most used are the conferences, the proceedings and the journal.

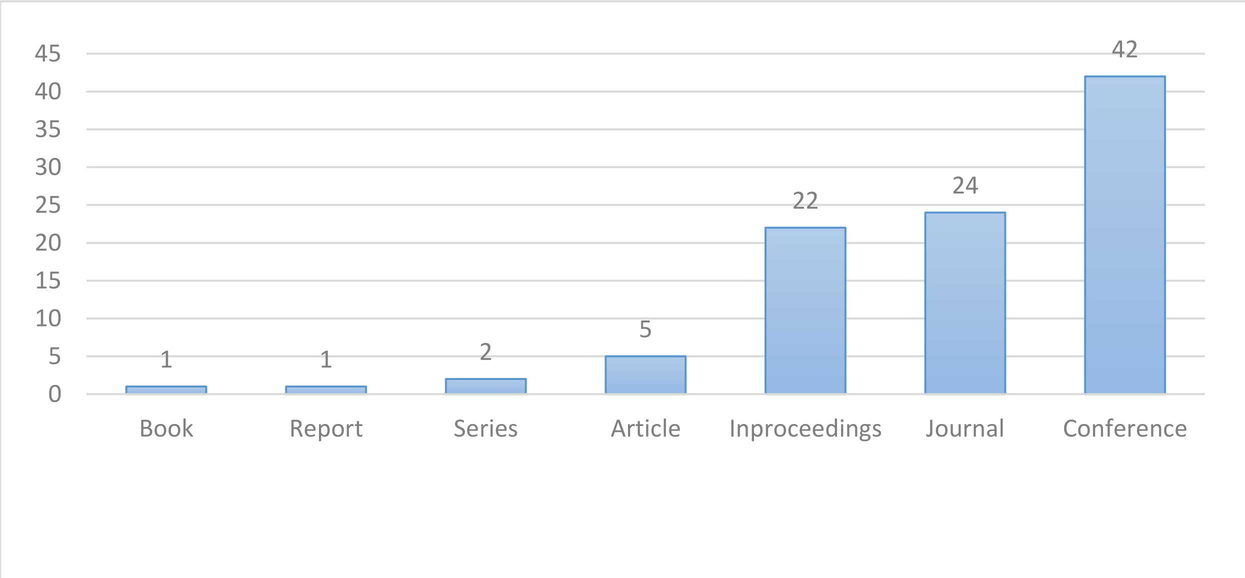

Figure 2-3 - Distribution of articles per type

2.2.4 REPORTING THE REVIEW

This section corresponds to the third and last step of the SLR methodology, where is summarized the extracted data from the selected documents.

2.2.4.1 DEVOPS ADOPTION

DevOps has a set of practices to make collaboration easier, continuous and fast delivery, reduce the lifecycle of development, development of high-quality systems, have a feedback loop [9,28,38,41] which means, feedback on every change [42,43] and mitigate the risk attached to the DevOps models, being this human or technological risk, [44,45] to respond to the customer needs [4].

The point of have DevOps teams is merge the Development and Operations team in only one so that it can lead to continuous integration and delivery [2,43]. This can only be done with the success of adoption of an agile culture [46]. The DevOps adoption focus on adopting new processes, skills, people behaviour and technological changes [46].

The demand of the today's world, have presented new needs to the software development, such as fast delivery, frequent change, low tolerance to failure, and fast adaption capability [7,28]. This way, companies have started to understand "why it is not possible to accelerate software projects by simply adding staff" [14].

DevOps, have showed organizations how to gain ruthless advantage [1], however there are some agile practices that must be considered, and identified [47], they are presented in Table (2-4). It is important to notice that agile approaches do not depend only on practices but also on the dynamism, culture, people, objectives, and size of the teams [48]. The adoption of DevOps can also depend on the project at hands [28,49,50].

The practices that are more referenced in the articles analyzed are continuous delivery, continuous integration, continuous deployment, continuous feedback, continuous test and automation, continuous improvement mindset, continuous learning, meeting daily to ease the communication of team members, prioritizing requirements in a backlog based on value. Although this are the practices that are more frequently found in the literature, there are others referenced in the Annexe 1.

With the major goal of being competitive, companies must deliver software rapidly [51]. For this, the development cycles must be shorter, this is where continuous delivery starts to be practice needed [51]. Continuous and fast delivery is an agile development practice, that aims to optimize the software engineering process, reduce the delay of delivery and costs that came with the delays, [52] it also enables on-demand deployment of the software developed in any development environment [1,15,38,42,48,53]. This practice, also named “release engineering” [54] is possible due to the collaboration between the development and operations teams [9,13]. The delivery can be done whenever is wanted, whether it is weekly or daily, depending on the company goal [55].

The Continuous Integration is very much related to the Continuous Delivery (the practice approached in the last paragraph) and the continuous testing (explained ahead in the document). Continuous Integration is a practice that consists in instantly integrate and test combination of the changes done to a software in an environment, in a way, it helps to apply quality control to the changes made in the process of software development [1,33,51,52,56–59]. This practice was created to help making the continuous delivery possible [2]. The goal of this practice is to eliminate barriers between the development and delivery of the features [7,37].

Continuous deployment is related to the Continuous Integration practice and consists in the release of the software product into a production environment [52] after the end of an agile sprint or when the continuous integration tests are passed [7]. With the Continuous deployment, there are low risk releases, and the development team can, more easily, adapt to requirements [3,52,58,60]. The adoption of this practice requires the effort of all the engineering team [55], so, automation of this deployments eases the embracement of this practice [33,57].

Continuous feedback aims to receive the feedback of all parties included in the process of the software development [1,33,41,61]. The fast and continuous feedback helps the development team trace the maintenance of the features being developed, trace the updates needed, the customer requirements and the clarification of any issue that requires attention. This way, problems can be detected before the development goes to production or impacts any customer business [6,7,42].

Continuous testing can be defined as “a software testing process which promotes test early and tests often” [15], which supports the rapid cycle of agile culture [37]. The principal purpose of this practice is to reduce the business risk, it includes quantitative and qualitative assessment for all risks and map how to mitigate those risks [15]. Although most companies do not have a mature test automation process, this practice has been more and more adopted [62]. Automation enters in the theme because it supports the continuous testing practice, allowing to ensure quality in the frequent releases [1,63]. It leads to Automated testing, which is one of the cornerstones of agile, by reducing the testing time, it allows companies to give to the customers a product with quality, without delaying its delivery [33,36].

This practice, the continuous improvement mindset, defends the hability to prioritize the quality and security, and still maintain the fast pace of delivery to production [1]. The continuous improvement is just possible if are measurements made continuously, so that it is possible to improve on the topics that improvement needs [1]. The continuous improvement depends on the ability of people to change, and DevOps is all about the collaboration between people and the agile mindset of the ones involved [42,49]. A factor that is also important on the improvement is the feedback of the users, the feedback of users is an engine for the improvement [7].

Being, DevOps, a culture that encourages the collaboration between teams and its members, the sharing of knowledge happens through all the development process, which means that the process of continuous learning arrives naturally [1,7,42,64]. Concerning that this collaboration between team members happens since the start of the projects, in the end, it will bring more value to the service and customers [38].

Software Development many times includes a lot of communication which bring us to a practice that is normal for Agile Development, meaning, to do a daily standup meeting to ease communication between team members [7,30,56].

Prioritizing requirements in a backlog is one practice of agile that is important since it helps to bring value to the customer and helps with communication [7,65]. Backlog is a list of stories to be developed and tested. This backlog must be redefined through time since the priorities change over the years [56].

The authors Snyder, B. Curtis say that “An average of 28% productivity gains was achieved by teams that implemented repeatable agile practices…” [66].

This way, agile practices can lead to a shift in the way that software development is developed and all of them are related between them [67,68]. So, there are some practices that enable and disable the agile adoption, these ones are approached in the next two Sections.

2.2.4.2 PRACTICES ENABLE AND DISABLE

With the successful agile adoption, companies can reach a level where they gain insight and make investments that can lead the organization to the right place [11].

In the Annex 1, are described some practices that can drive companies to successfully adopt agile culture, these, are divided in two groups, cultural and technological [46].

The agile practices, them being, collective ownership of shared values (all team working together have the same values and all adopt the same practices), the definition of success to be equal to all people working together, the communication between all team members involved to be easy so that the learning and shared experiences becomes continuous, give incentives to all teams to keep them motivated and focused, implement the culture of respect, trust, responsibility, set shared goals, mindset and way of working. To the technical enablers is added, to build automation (deployment, infrastructure, monitoring automation, recovery and test), configuration management [9,46,49].

In Anexx 3, are some factors that act as impediments that must be removed because they drive organizations in the opposite path of agility [11].

Some of the factors that cause barriers on the adoption of an agile culture are, communication and bureaucratic barriers [1,3,15,24,30,69], geographical distribution [30,46,70,71], lack of education around DevOps [3,46,71,72], cultural changes [3,28,70], differentiation of processes for the different countries [69–71].

2.2.4.3 BENEFITS OF DEVOPS IN AGILE SOFTWARE DEVELOPMENT

The eminent change in competition, technology and complexity of softwares has led to an increase of necessity related to rapid, reliable, scalable, and evolved processes [14]. Pursuit the business success is the reason why many organizations try this agile transformation [11].

DevOps leads to the elimination of issues like the lack of collaboration between the team of operations and development, by this adjoined teams it is possible to promote cooperation, collaboration, and communication [25,33]. The benefits of DevOps do not end here and so there is Annex 4, where are some of the benefits are presented by the articles analysed.

Some of the benefits that are more referred in the articles are, the faster feedback [1,6,7,33,41,42,61,73], collaborative work [13,24,42,43,74], faster fixes [31,43,51,75], reduce the technological and human risk [9,14,45,75], multidisciplinary and self-improving teams [13,30,43], features arrive faster [9,60,75] and at the end, as a consequence of all of this benefits, the change happens faster [14,42,51].

2.2.4.4 PERFORMANCE KPI'S

Adopting an agile culture, allows the more frequent release of reliable software [71,76], this happens due to the speeding of the software development [14]. The change that the adoption of agile culture demands is in all the organization [46] and a resilient performance depends on the capacity of the organization to make that change.

The today's world makes a demand for resilient performance and so it is important to try to measure it [16]. With this, enters the Table 2-4, where are the KPI's to measure the performance of DevOps teams.

Table 2-4 – KPI's for DevOps teams' measurement

Type	KPI	Description	Articles
Delivery	KPI1	Time defined to end project - Actual time to end project	[39,40,77]
	KPI2	ITV= (Nº of developers/How many days are in one interaction) *(How long is the iteration in business days)	[40]
	KPI3	RTV= Nº of development iterations * ITV	[40]
	KPI4	Number of days a requirement is behind its deadline	[39]
Costs	KPI5	Labour costs planned - Actual labour costs	[40]
	KPI6	Project costs planned – Actual project costs	[40]
Defects	KPI7	Manpower effort defects / Manpower effort project	[40]
	KPI8	Time to Failure	[40,71]
	KPI9	Number of higher priority incidents within the first six weeks after release.	[39,77]
	KPI10	Higher priority defects / Effort of a system release.	[39,77]
	KPI11	Number of all open defects by their priority.	[39,78]
Tests	KPI12	(Number functionalities Tested*100)/Total Number Functionalities	[39,40,77,78]
	KPI13	(Tests Passed|Failed|Broken*100)/ Total number of tests	[39,40,77,79,80]

The first one (KPI1) is doing the difference between the ideal and actual time for ending a project, this measures the timely completion of a project milestone, comparing this measure in different times, will help to determine the performance of a team [39,40,77].

There are some indicators to do project measurements, by measuring the project it is possible to judge the performance of the team, to this there are the KPI2 and KPI3, to measure the Iteration Team Velocity (ITV) and Release Team Velocity (RTV), respectively. The ITV is measured dividing the number of developers by how many days there are in one interaction and multiplying that for how long the iteration in business days is. The RTV is calculated by multiplying the number of developers for the ITV. With these KPI's it is possible to try and calculate the ideal number of days per interaction and release, the ITV gives the ideal days of an iteration, the RTV give the days, based on the ITV, that a release must take. These values can, later, be analysed and help determine the performance of the team [40].

The delay of requirements is another scope that must be analysed (KPI4) by taking in consideration this, it is possible to identify the delay on developing requirements (and features) that can help assess the risk of reduced software quality and consequently, performance of the team [39].

The KPI5, proposed the difference between Labor Costs Planned and Actual Labor Costs, this KPI gave to the top management the ability to measure the manpower effort, being the ration of the costs described [40]. In here there is also KPI6 that allows to analyse if the project stayed within the budget done initially and for how much [40].

There are KPI for measure of development quality (KPI7) this is the quotient between the Manpower Effort to result defects and the Manpower Effort total of the project. This gives a notion of the manpower used to resolve some defects of the development previously done. This KPI can be used in different stages of the project so that is possible to understand if this number is reducing or not [40].

The KPI8, points to the mean time to failure, is a reliability indicator that gives a notion in how reliable a development is being done by a team [40,71].

The number of higher priority incidents (KPI9) can “measure the quality of delivered systems in terms of user feedback regarding higher priority incidents” [39], this KPI helps understand the quality of the software delivered, since this same quality can reflect the team’s performance [77].

It can also be considered the priority defects per person a day by the KPI10, measuring the ratio of higher priority defects to the development effort, allows to compare the quality and costs of different releases of a system with each other, making it possible to see the performance of a team by comparing results for the projects [39,77].

Measuring the number of all open defects in a system distinguished by their priority (KPI11) it is possible to measure the quality of a project related to defects and higher priority defects [39]. By this, it is possible to interpretate the team’s performance by understanding the quality of the project developed by them. In the scope of software and technical quality takes the defect density and defect removal efficiency to try to determine and interpretate the team’s performance [40,77].

The percentage coverage of the functionality by test cases (KPI12) gives a notion of the amount of testing performed by a set of tests, which can give the quality of the testing being done by the team [76]. The bigger the percentage, the more functionality is being covered by the tests and better is the performance of the team in this scope [39,40,77]. The KPI13 is also related with the testing, and measures the percentage of tests passed, failed, or broken. Once more, the bigger this percentage is, the best, because it means that the development being done has no failing points [39,40,77,80]. The measurement and analysis of the tests made can give an indication of the reliability of the work being tested [76].

3 RESEARCH METHODOLOGY

Since this investigation aims to find factors that influence the performance of the DevOps teams, the goal of the research is exploratory in nature. For the next phase of this SLR, an exploratory CS will be performed to apply the set of KPI's to a DevOps team and try to analyse the variations of the KPI's results and the factors that can explain those changes.

A CS can be performed for many purposes. In the case of this investigation, a case study is being performed to try to find factors that can be improved [81]. The point is to use the KPI's found in the SLR and apply them to a real DevOps team. With the data received, factors will hopefully be able to explain the results and possible variations of the KPI's and "develop a comprehensive model describing patterns of behaviour" [81].

As mentioned before in this investigation, there are very few empirical studies about this topic. Therefore, an Exploratory CS will be performed. An "Exploratory research is meant to start a study on a determined phenomenon observed, where there are no prior (or few) works" [82].

A DevOps team willing to apply the metrics was selected. Since the point is to find the KPI's that make sense to calculate to the specific team and apply them and, if needed, question the team members about the oscillations in the results. The interviews and questions to the team members, were depending on the results of the KPI's.

The team under study was created in January 2022 and composed of fifteen people, eight Developers (four Portuguese and four Indian), one Business Analyst (BA), two Architects, and four Product Owners (PO). The team also has a team leader that is one of the PO´s.

The DevOps team analysed is a team that is distributed in the globe, four of the developers are working from Portugal and four are working from India, where 2 are girls and 6 are men. The Portuguese developers are internal in the organization and have already worked for the same company before January of 2022 but in another department that were set apart. The Indian developers are external colleagues that also already worked for the company before the creation of the team. For the PO's, 3 of them are from Portugal and 1 is from Germany, all of them already worked for the team before January 2022. The team leader is one of the PO's that are from Portugal and already worked in the company before coming to this team but is the first time with the position of team leader. For the two architects, one of them is from Portugal and the other is from Germany, both also worked for the team.

The team works in two-week sprints where the development of stories is done plus one week for review of the stories before the sprint starts. The weekly review consists of analyse all the requirements requested in the stories and give points that represents the effort (time) needed to implement. Depending on the team's capacity (in hours) for the sprint, the stories are included on the sprint. Today, the team works in three different products.

It is important to mention that when the team was put together, they started a parallel project while they worked. The team was working in different teams in the company, but all for a major project in which they already had their instances put together, where they developed all the new requirements from the customer. But in January of 2022, then the team was put together, it was with the promise that they would leave the instance approximately 6 months from there. This way, the team started to work in parallel to give the customer new requirements but also put together their own instances with all the functionalities that the customers already had in the old ones.

The team divides the stories in 3 types, feature, production issues and maintenance stories. The first ones are stories that intent to add new functionalities, production issues represent some issues in the production environment that were introduced by the release of new features, maintenances are stories that intent to do little maintenance tasks to features that already exist. Every week exists a release that takes the new developments from the development environments to production so that the users can have access to the new functionalities.

The investigation starts by choosing which sprints were going to be analysed. With the objective of having data to compare, 3 sprints were chosen to analyse. To choose the sprints, was important to have in consideration that, as the team was new, there had to be time to find their way of working together before starting to analyse them. So, the first sprint chosen was two months after the team started to work together and the others are sequential to the first. Then the KPI's were applied to each one of the sprints.

The second step was to analyse the data that was needed to fulfil the KPI's and the data that was provided by the company. In this point was found that some of the data needed to calculate the KPI's was not permitted access by the company and other was not recorded or not applied to the team. In this point was discovered which KPI's were possible to calculate and applied to the team way of work. It was then provided access to all the necessary documentation like, sprints duration, stories duration and stories independent history.

4 CONDUCTING THE CASE STUDY

4.1 CASE STUDY

The KPI's were calculated individually for each sprint. The sprint data can be found resumed in the next bar graphic (Figure 4-1).

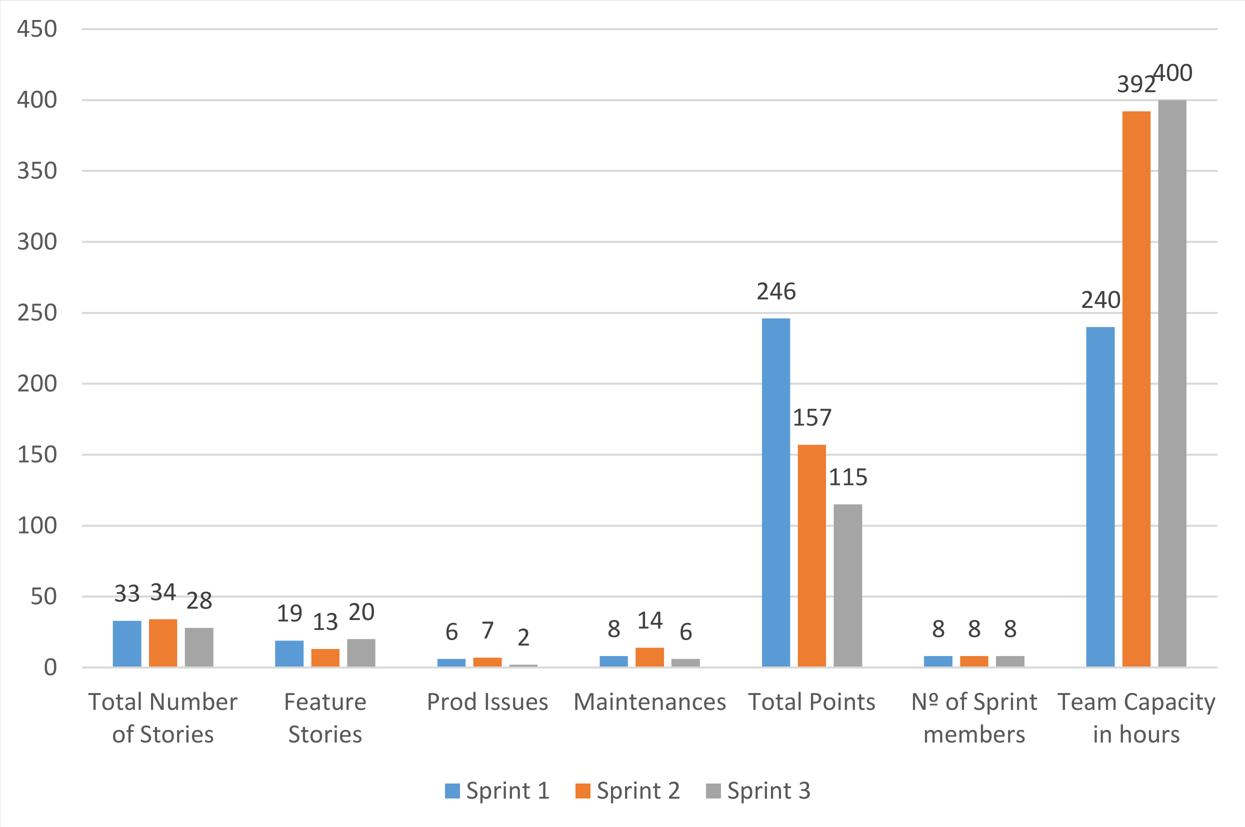

Figure 4-1 - Sprint Data Resume

From the list of KPI's found, some of them were not possible to calculate those are, KPI1, KPI2, KPI3, KPI5, KPI6, KPI8 and KPI13. The KPI1 and KPI8 were not possible to calculate due to the three projects in which the team observed works, are long term projects, what means that they have no end date. For KPI2 and KPI3, those do not have a result because the article in which The KPI's were documented, does not have enough documentation about the way this metric is used and the result that it translates to us. Both KPI5 and KPI6 are measures related to costs, the company in which this team works, did not gave us access to data related to costs, so this KPI's were not calculated. KPI13 is related to the tests passed, failed, or broken but since this team does not have a tester or a team of testers incorporated, the ones testing the story are the developers and the POs. This way, the team does not take a track of all the tests done in one story.

4.2 ANALYSIS OF THE SPRINTS

4.2.1 SPRINT 1

This sprint counted with 33 stories, 19 feature stories, six production issues and eight maintenances, on total which all together had a total of 246 points. The team reached a capacity of 420 hours for the sprint. Which means that the team capacity is close to 1.71 times higher than what the sprint needs.

For the KPI4, the number of days a requirement is behind the deadline, nine requirements (stories) were delayed. The minim days were five and the maximum 36. Stories 1, 4 and 6 were both delayed due to its implementation depended on other team's work that was not yet developed and tested. To Story 2, the reason of the delay was that the story needed to wait for the approval of the customer and not only for the approval of the PO. Story 3 was delayed due to pending decisions, that were raised during the development of the story, from the POs. Story 5 was delayed because during the development, it was noticed that the solution that was thought by the team, was impacting other developments done in the instance, this obligated the team to discuss a new solution that did not impact any other feature. For story 7, the delay on the conclusion of the story was due to the requirement not being 100% clear and so, other story was created to address some topics that were not clear in the initial description of the story, after this, there was a dependency between the two stories, and this story was only completed to be deployed in production after the second story was also completed. Story 8 e 9 were both delayed due to the time that the PO spent to test the new functionality/feature.

The KPI7 measures that 38% of team is occupied with the production issues. For KPI9, the result is 0 since there was no higher priority incidents registered after the releases. KPI10 tell us that 25% of the team was concerned with prod issues that had higher priority. KPI11 tells us the number of defects opened by their priority, there are two with priority one, two with priority three and two with priority four. The lowest priority means higher impact on the product/system.

For the KPI12, it's possible to analyze the percentage of functionalities/features that were tested in the sprint, on this case, 100% of the functionalities added or modified were tested.

Table 4-1 - Results Sprint 1

Type		
Delivery	KPI4	STRY1 - 30 days STRY2 - 11 days STRY3 - 34 days STRY4 - 30 days STRY5 - 5 days STRY6 - 30 days STRY7 - 6 days STRY8 - 36 days STRY9 - 34 days
Defects	KPI7	3/8 = 0,38
	KPI9	0
	KPI10	2/8 = 0,25
	KPI11	Priority 1= 2 Priority 3= 2 Priority 4= 2
Tests	KPI12	100,00

4.2.2 SPRINT 2

The second sprint counted with 34 stories, 13 feature stories, seven production issues and 14 maintenances, on total which all together had a total of 157 points. The team reached a capacity of 392 hours for the sprint.

For the first KP4, there are 11 stories that were delayed. The reason for the delay of the first story was because it needed to be approved and tested by the customers. Story 2 was delayed since it was depending on the new instances that the team were preparing to go live, only being completed once it happened. For story 3, 6, 7, 8, 9 and 10, the PO spent more time to test the story than the defined time. Story 4 was delayed due to the time the PO and the developer waited for the required translations, being this tardiness caused by a third-party dependency. Story 5 was a maintenance story that was added to the sprint in its last day and the developer had no time to start it before the end of the sprint. For story 11, the delay happened because the story failed the tests, and the developer could not finish the story in the time of the sprint.

The KPI7 measures that 63% of team is occupied with the production issues. For KPI9, the result is 0 since there was no higher priority incidents registered after the releases. KPI10 tell us that 25% of the team was concerned with production issues that had higher priority. KPI11 tells us the number of defects opened by their priority, there are two with priority one, two with priority two and one for priority three, four and five.

For the KPI12, it is possible to analyze the percentage of functionalities that were tested in the sprint, for this case, 94,12% of the functionalities added or modified were tested.

Table 4-2 - Results of Sprint 2

Type	**KPI**	
Delivery	KPI4	STRY1- 15 days STRY2- 100 days STRY3- 21 days STRY4- 20 days STRY5- 6 days STRY6- 19 days STRY7- 22 days STRY8- 1 days STRY9- 19 days STRY10- 19 days STRY11- 95 days
Defects	KPI7	5/8 = 0,63
	KPI9	0
	KPI10	2/8 = 0,25
	KPI11	Priority 1= 2 Priority 2= 2 Priority 3= 1 Priority 4= 1 Priority 5= 1
Tests	KPI12	(32*100) / 34 = 94,12

4.2.3 SPRINT 3

This sprint counted with 28 stories, 20 feature stories, two production issues and six maintenances, on total which all together had a total of 115 points. The team reached a capacity of 400 hours for the sprint.

The current sprint being analysed counts with 13 stories being delayed. For the story 1, 2, 3, 4, 5, 8, 10, 11, 12 and 13 the reason of the delay was the PO not testing the story in the time defined. The stories 6 and 7 were both delayed due to the dependency between them, caused by the necessity of customer approval of the new feature. Story 9 was also delayed not due to the PO but due to the approval of the customer.

The KPI7 measures that 25% of team is occupied with the production issues. For KPI9, the result is 0 since there was no higher priority incidents registered after the releases. KPI10 tell us that 0% of the team was concerned with production issues that had higher priority. KPI11 tells the number of defects opened by their priority, there are two with priority three.

For the KPI12 it is possible analyze the percentage of functionalities that were tested in the sprint, for this case, 96,43% of the functionalities added or modified were tested.

Table 4-3 - Results Sprint 3

Type	KPI	
Delivery	KPI4	STRY1- 3 days STRY2- 4 days STRY3- 1 days STRY4- 45 days STRY5- 5 days STRY6- 3 days STRY7- 3 days STRY8- 6 days STRY9- 5 days STRY10- 50 days STRY11- 51 days STRY12- 57 days STRY13- 56 days
Defects	KPI7	2/8 = 0,25
	KPI9	0
	KPI10	0/8 = 0
	KPI11	Priority 3= 2
Tests	KPI12	(27*100) / 28 = 96,43

4.3 FINDINGS AND PROPOSALS

After analysing the KPI’S for each sprint individually, it is important to resume the findings so that it is possible to get more accurate conclusions. So, to help conclude the findings of the previous analysis, there is Figure 4-2.

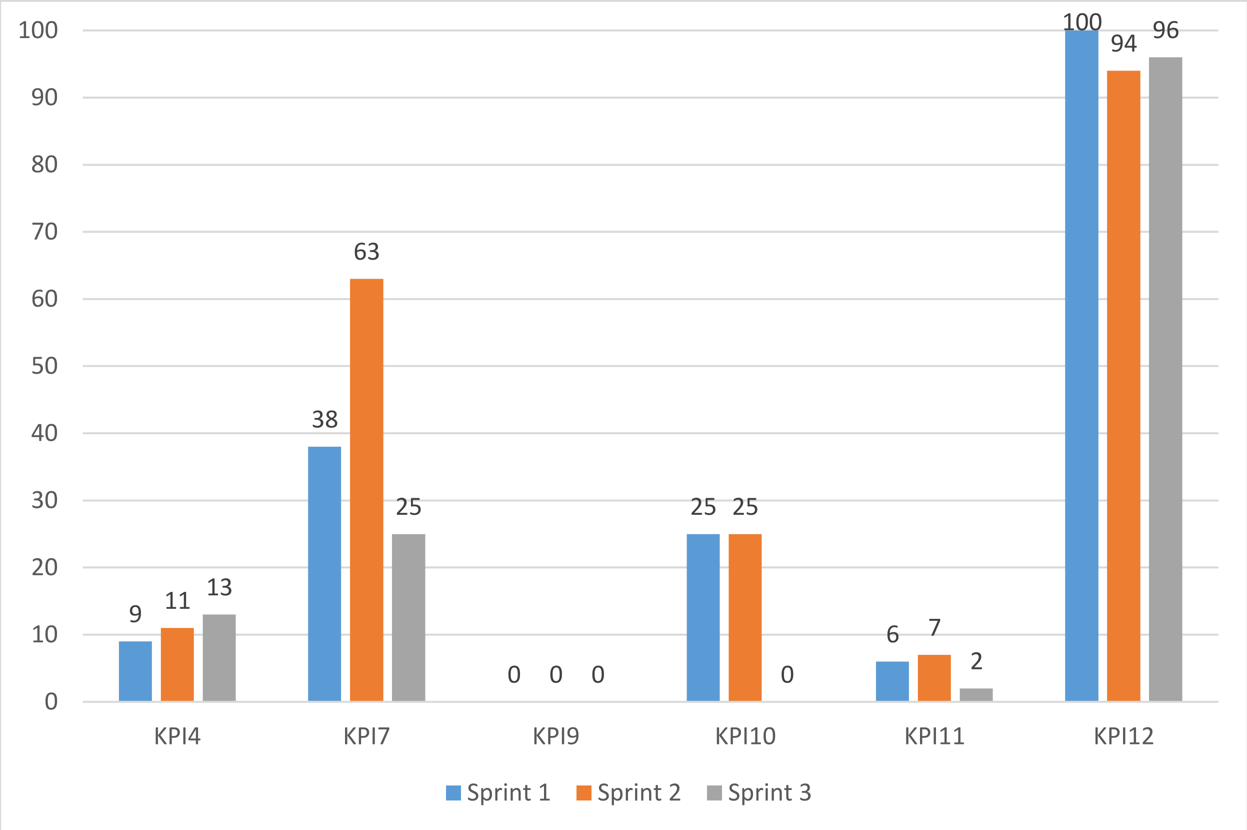

Figure 4-2 - Sprint Findings Resume

For KPI4, it is possible to see that the number of stories with delay has increased from one sprint to the other. About the reason that led to this delay, it is possible to see in Table 4-4, a list of the reasons that explain this lateness. According to Table 4-4, the reasons that are more common for the delay of stories are the lateness on the PO's testing, the need for approval of the customer, the dependency on other teams and some pending decisions that are not discussed in the review of the analysis of the requirements on the review of the stories.

Table 4-4 - Delay Reasons (KPI4)

Reason	Number of stories with this delay reason
PO spent more time then estimated to test the new functionality.	18
Needed of customer approval.	5
Implementation has third party dependencies.	3
Pending decisions, that raise during the development of the story.	2
The requested requirements have impact in other implemented features	1
Requirement is not 100% clear.	1
Story was waiting for testing in the new instances	1
Developer did not have the time, in the sprint, to develop the story.	1
Story failed the tests	1

For KPI7, it’s possible to see the percentage of team that is occupied, during the sprint, with production issues. For that point, it’s possible to analyze that this number has increased in sprint 2 and decreased in the next sprint. This behavior can be explained by the number of production issues that appear in the sprints, in sprint 2 this number was higher but in sprint 3, it decreased again.

KPI9 does not tell us much because there was no registry of any higher priority incidents in any of the sprints. This information just tells us that the team is doing a good job in working to avoid this type of incidents.

KPI10, lets us know the percentage of the team that was occupied with high priority production issues. For this KPI is possible to conclude that just 25% of the team is occupied with production issues of priority one, in the first and second sprint. For the third sprint, this value is 0 since there weren’t registered production issues with priority one, as can be seen in Figure 4-3.

KPI11 presents the number of defects opened by priority. For a better understanding of the values of this KPI, there is Figure 4-3. So, comparing the three sprints, it is possible to see that the number of production issues with priority one, is the same in the first two sprints and in the third it decreased to zero. For priority two, just the second sprint counted with two of those. For priority three, the number of production issues with that priority decreased in 1 and in the second sprint and the third, this number increased again in 1. For priority four, the behaviour is the same to the ones with priority one. To priority five, just the second sprint registered one production issue.

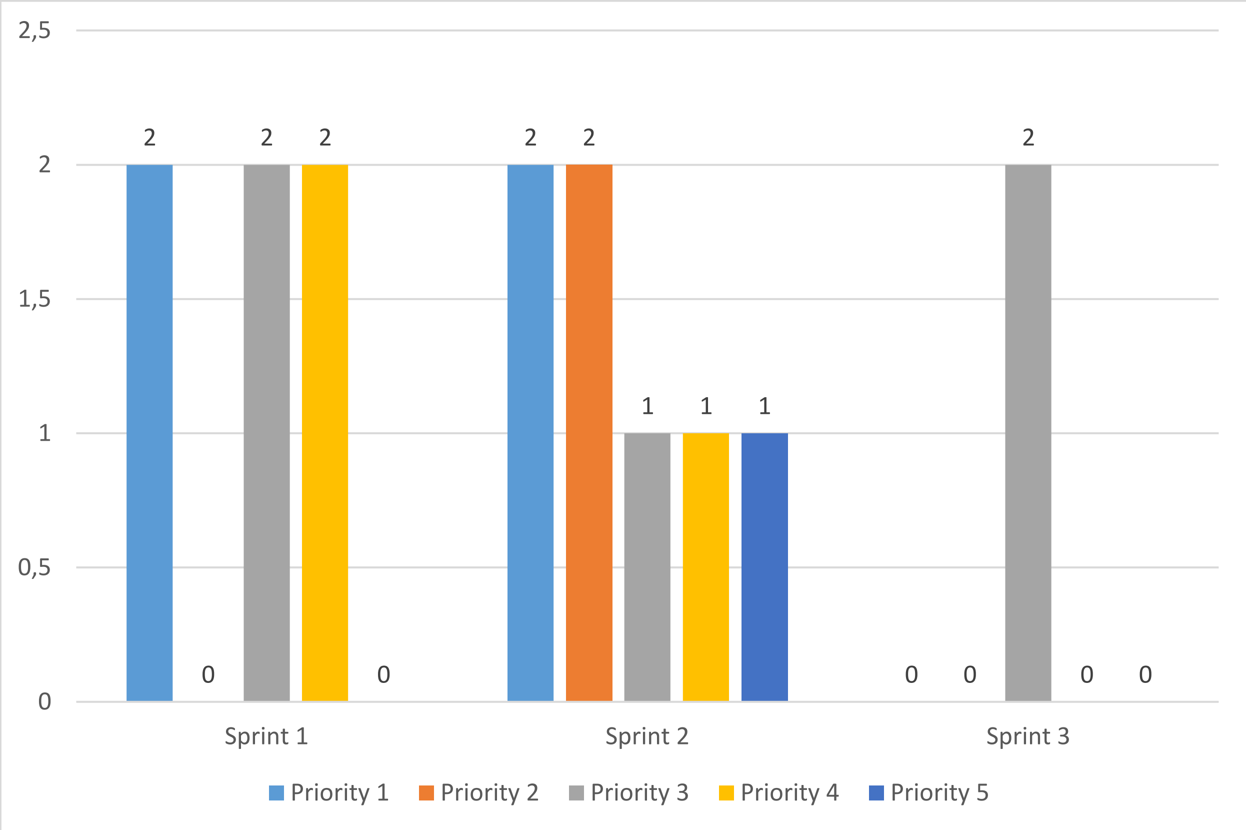

Figure 4-3 - Production Issues by priority

For the KPI12, it's possible to analyze the percentage of functionalities that were tested in the sprint. According to the Figure 4-2, just the first sprint counted with 100% of the functionalities tested. In the second sprint this number decreased to 94% and in the third this number increased 2%. Although this number is not very high, it is important to understand why this percentage decreased on sprint 2 and 3. Translating this percentage to numbers, in the second sprint, two stories were not tested and in the third, just 1 story was not tested. One of the stories of the second sprint were not tested because the POs were waiting for the new instances to test the feature. For the second story in this sprint, it was not tested because its implementation depended on another story development and that story also had dependencies on other stories that were also not tested by the PO. For the story that was not tested in the third sprint, the PO spent so much time to test it that the Business Analyst of the team, found an issue with the platform itself that it ended up messing with the implementation of this story, this way, a case was open and was still waiting for an answer from the support team of the application, and for this reason this story was blocked and was never tested by the PO.

It’s possible to analyse some issues and maybe take some measures so that some of the KPI’s can have better results. Some of the issues are:

- The number of stories delayed have been increasing over sprints,
  - PO spent more time then estimated to test the new functionality,
  - Need of customer approval,
  - Implementation has third party dependencies,
  - Pending decisions, that were raised during the development of the story,
- The number of production issues is not very high, but it should be analysed if it can be reduced
- The number of stories tested i only 100% in the first sprint, no stories should be left without testing.

To the points that were presented, some measures can be taken, as:

1. Take the POs test the stories in the period pretended with the penalty of not being able to open stories in the next sprint,
2. When a story is included in the sprint, guarantee that there are no pending decisions, if there is, the story should not enter the sprint until any decision is left to make,
3. Make sure that the third-party dependencies are settled allowing the story implementation. If these conditions are not met, the story should be blocked until all the pre-requirements are set,
4. The number of production issues can be reduced by adding a tester to the team. This would also help with the high number of stories left to test in a sprint.
5. Another measure that would help the team to be more focused on new features would be to add an Operation Team, focused on operations (dealing with production issues and maintenances).

5 CONCLUSION

DevOps, being behind the automation, continuous processes, and collaboration between people [11], but it cannot improve the performance of teams and consequently organizations on its own. There are practices that must be taken in consideration, the practices adopted differ from team to team and organization to organization, people and environments are not the same.

It is possible to say, after this research, that those KPI’s referred to testing quality and quality development are more frequent found in articles. However, it is important, when measuring KPI’s to take in consideration the defects and incidents, to make sure, these are really teams’ fault and can be allocated to them.

The KPI’s can help companies to well informed decisions about the teams and projects as well as identify business opportunities. Although companies found the importance of the KPI’s to measure performance of agile teams, this is a matter, yet little explored by the community.

The major limitation of this work is the lack of KPI’s found in the research. Although there are many articles about DevOps practices, benefits, and the importance of agile measurement, there is a lack of KPI’s with focus on measuring teams’ performance. Another limitation is the different interpretation of KPI’s, the meaning of KPI is different for many people and businesses.

Seeing this, a potential future work on this matter, is to do a MLR to try to find more information in grey literature (this can be, non-conventional, non-commercial, and semi-published publications) about the KPI’s since is already talked about performance and its importance but not concerning team’s scope.

6 REFERENCES

[1] M. Sánchez-Gordón, R. Colomo-Palacios, Security as Culture: A Systematic Literature Review of DevSecOps, in: Proc. IEEE/ACM 42nd Int. Conf. Softw. Eng. Work., Association for Computing Machinery, New York, NY, USA, 2020: pp. 266–269. https://doi.org/10.1145/3387940.3392233.

[2] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, T. Zimmermann, Software Engineering for Machine Learning: A Case Study, in: 2019 IEEE/ACM 41st Int. Conf. Softw. Eng. Softw. Eng. Pract., 2019: pp. 291–300. https://doi.org/10.1109/ICSE-SEIP.2019.00042.

[3] J. Cito, F. Oliveira, P. Leitner, P. Nagpurkar, H.C. Gall, Context-based analytics - establishing explicit links between runtime traces and source code, in: 2017 IEEE/ACM 39th Int. Conf. Softw. Eng. Softw. Eng. Pract. Track, 2017: pp. 193–202. https://doi.org/10.1109/ICSE-SEIP.2017.1.

[4] S. Abdelkebir, Y. Maleh, M. Belaissaoui, An Agile Framework for ITS Management In Organizations: A Case Study Based on DevOps, in: Proc. 2nd Int. Conf. Comput. Wirel. Commun. Syst., Association for Computing Machinery, New York, NY, USA, 2017. https://doi.org/10.1145/3167486.3167556.

[5] S. Rafi, W. Yu, M.A. Akbar, A. Alsanad, A. Gumaei, Multicriteria Based Decision Making of DevOps Data Quality Assessment Challenges Using Fuzzy TOPSIS, IEEE Access. 8 (2020) 46958–46980. https://doi.org/10.1109/ACCESS.2020.2976803.

[6] A.F. Nogueira, J.C.B. Ribeiro, M.A. Zenha-Rela, A. Craske, Improving La Redoute's CI/CD Pipeline and DevOps Processes by Applying Machine Learning Techniques, in: 2018 11th Int. Conf. Qual. Inf. Commun. Technol., 2018: pp. 282–286. https://doi.org/10.1109/QUATIC.2018.00050.

[7] M.P. Barcellos, Towards a Framework for Continuous Software Engineering, in: Proc. 34th Brazilian Symp. Softw. Eng., Association for Computing Machinery, New York, NY, USA, 2020: pp. 626–631. https://doi.org/10.1145/3422392.3422469.

[8] Sharpening the Board's Cybersecurity Acumen., NACD Dir. (2019) 6–10. https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=137606590&lang=pt-pt&site=ehost-live&scope=site.

[9] V. Zamfir, M. Carabas, C. Carabas, N. Tapus, Systems Monitoring and Big Data Analysis

Using the Elasticsearch System, in: 2019 22nd Int. Conf. Control Syst. Comput. Sci., 2019: pp. 188–193. https://doi.org/10.1109/CSCS.2019.00039.

[10] E. Buis E.G., S. Ashby S.R., K. Kouwenberg K.P.A., Increasing the UX maturity level of clients: A study of best practices in an agile environment, Inf. Softw. Technol. 154 (2023) 107086. https://doi.org/https://doi.org/10.1016/j.infsof.2022.107086.

[11] M. Laanti, Agile Transformation Model for Large Software Development Organizations, in: Proc. XP2017 Sci. Work., Association for Computing Machinery, New York, NY, USA, 2017. https://doi.org/10.1145/3120459.3120479.

[12] D. Teixeira, R. Pereira, T. Henriques, M.M. da Silva, J. Faustino, M. Silva, A maturity model for DevOps, Int. J. Agil. Syst. Manag. 13 (2020) 464–511. https://doi.org/10.1504/IJASM.2020.112343.

[13] M. Michel, Managing Technical Debt in the Data Warehouse., Bus. Intell. J. 22 (2017) 24–32. https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=123029761&lang=pt-pt&site=ehost-live&scope=site.

[14] D. Rosenberg, B. Boehm, B. Wang, K. Qi, Rapid, Evolutionary, Reliable, Scalable System and Software Development: The Resilient Agile Process, in: Proc. 2017 Int. Conf. Softw. Syst. Process, Association for Computing Machinery, New York, NY, USA, 2017: pp. 60–69. https://doi.org/10.1145/3084100.3084107.

[15] J. Angara, S. Prasad, Continuous testing real-time health analytics dashboard, Int. J. Adv. Trends Comput. Sci. Eng. 9 (2020) 1713–1719. https://doi.org/10.30534/ijatcse/2020/123922020.

[16] D.D. Woods, J. Allspaw, Revealing the Critical Role of Human Performance in Software, Commun. ACM. 63 (2020) 64–67. https://doi.org/10.1145/3380468.

[17] S.D.G. Avila, P.O. Cano, A.M. Mejia, I.S. Moreno, A.N. Lepe, G.E.Z. Dominguez, A data analysis platform to evaluate performance during software development process, RISTI - Rev. Iber. Sist. e Tecnol. Inf. 2020 (2020). https://doi.org/10.17013/risti.36.50-64.

[18] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman, Systematic literature reviews in software engineering – A systematic literature review, Inf. Softw. Technol. 51 (2009) 7–15. https://doi.org/https://doi.org/10.1016/j.infsof.2008.09.009.

[19] Z. Zainal, Case study as a research method, J. Kemanus. 9 (2007).

[20] J. Díaz, J.E. Pérez, M.A. Lopez-Peña, G.A. Mena, A. Yagüe, Self-Service Cybersecurity Monitoring as Enabler for DevSecOps, IEEE Access. 7 (2019) 100283–100295. https://doi.org/10.1109/ACCESS.2019.2930000.

[21] L. Prates, J. Faustino, M. Silva, R. Pereira, DevSecOps Metrics, in: S. Wrycza, J. Maślankowski (Eds.), Inf. Syst. Res. Dev. Appl. Educ., Springer International Publishing, Cham, 2019: pp. 77–90.

[22] M.A. Silva, J.P. Faustino, R. Pereira, M.M. da Silva, Productivity gains of DevOps adoption in an IT team: A case study, in: Proc. 27th Int. Conf. Inf. Syst. Dev. Des. Digit. ISD 2018, 2018. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85083567784&partnerID=40&md5=d4c85310ac05a346410cfd3ac0b15e21.

[23] D. Renzel, I. Koren, R. Klamma, M. Jarke, Preparing Research Projects for Sustainable Software Engineering in Society, in: 2017 IEEE/ACM 39th Int. Conf. Softw. Eng. Softw. Eng. Soc. Track, 2017: pp. 23–32. https://doi.org/10.1109/ICSE-SEIS.2017.4.

[24] S.W. Hussaini, Strengthening harmonization of Development (Dev) and Operations (Ops) silos in IT environment through systems approach, in: 17th Int. IEEE Conf. Intell. Transp. Syst., 2014: pp. 178–183. https://doi.org/10.1109/ITSC.2014.6957687.

[25] U. Van Heesch, T. Theunissen, O. Zimmermann, U. Zdun, Software Specification and Documentation in Continuous Software Development: A Focus Group Report, in: Proc. 22nd Eur. Conf. Pattern Lang. Programs, Association for Computing Machinery, 2017. https://doi.org/10.1145/3147704.3147742.

[26] A. Bulgarelli, The AGILE Gamma-Ray observatory: software and pipelines: Software management approach and lessons learned for the next generation of high-energy observatories., Exp. Astron. 48 (2019) 199–231. http://10.0.3.239/s10686-019-09644-w.

[27] D. Teixeira, R. Pereira, T.A. Henriques, M. Silva, J. Faustino, A systematic literature review on DevOps capabilities and areas, Int. J. Hum. Cap. Inf. Technol. Prof. 11 (2020) 1–22. https://doi.org/10.4018/IJHCITP.2020070101.

[28] D. Jana, P. Pal, ESSENCE Kernel in Overcoming Challenges of Agile Software Development, in: 2020 IEEE 17th India Counc. Int. Conf., 2020: pp. 1–8. https://doi.org/10.1109/INDICON49873.2020.9342375.

[29] S. Martínez-Fernández, A.M. Vollmer, A. Jedlitschka, X. Franch, L. López, P. Ram, P. Rodríguez, S. Aaramaa, A. Bagnato, M. Choraś, J. Partanen, Continuously Assessing and Improving Software Quality With Software Analytics Tools: A Case Study, IEEE Access. 7 (2019) 68219–68239. https://doi.org/10.1109/ACCESS.2019.2917403.

[30] A. Patwardhan, Sentiment Identification for Collaborative, Geographically Dispersed, Cross-Functional Software Development Teams, in: Proc. - 2017 IEEE 3rd Int. Conf. Collab. Internet Comput. CIC 2017, 2017: pp. 20–26. https://doi.org/10.1109/CIC.2017.00014.

[31] M.Á. Conde, A. Sarasa-Cabezuelo, J.-L. Sierra, 9th International Workshop on Software Engineering for ELearning (ISELEAR'18), in: Proc. Sixth Int. Conf. Technol. Ecosyst. Enhancing Multicult., Association for Computing Machinery, New York, NY, USA, 2018: pp. 879–882. https://doi.org/10.1145/3284179.3284327.

[32] E. Kanoulas, J. Karlgren, Practical Issues in Information Access System Evaluation, SIGIR Forum. 51 (2017) 67–72. https://doi.org/10.1145/3130332.3130344.

[33] M. Zaydi, B. Nassereddine, DevSecOps PRACTICES FOR AN AGILE AND SECURE IT SERVICE MANAGEMENT., J. Manag. Inf. Decis. Sci. 22 (2019) 527–540. https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=142289426&lang=pt-pt&site=ehost-live&scope=site.

[34] M. Kalinowski, S.T. Batista, H. Lopes, S. Barbosa, M. Poggi, T. Silva, H. Villamizar, J. Chueke, B. Teixeira, J.A. Pereira, B. Ferreira, R. Lima, G. da S. Cardoso, A.F. Teixeira, J.A. Warrak, M. Fischer, A. Kuramoto, B. Itagyba, C. Salgado, C. Pelizaro, D. Lemes, M.S. da Costa, M. Waltemberg, O. Lopes, Towards Lean R&D: An Agile Research and Development Approach for Digital Transformation, in: 2020 46th Euromicro Conf. Softw. Eng. Adv. Appl., 2020: pp. 132–136. https://doi.org/10.1109/SEAA51224.2020.00030.

[35] H. Sukhwani, R. Matias, K.S. Trivedi, A. Rindos, Monitoring and Mitigating Software Aging on IBM Cloud Controller System, in: 2017 IEEE Int. Symp. Softw. Reliab. Eng. Work., 2017: pp. 266–272. https://doi.org/10.1109/ISSREW.2017.65.

[36] O. Liechti, J. Pasquier, R. Reis, Supporting Agile Teams with a Test Analytics Platform: A Case Study, in: 2017 IEEE/ACM 12th Int. Work. Autom. Softw. Test., 2017: pp. 9–15. https://doi.org/10.1109/AST.2017.3.

[37] H. Chen, R. Kazman, S. Haziyev, Agile Big Data Analytics for Web-Based Systems: An

Architecture-Centric Approach, IEEE Trans. Big Data. 2 (2016) 234–248. https://doi.org/10.1109/TBDATA.2016.2564982.

[38] A. Balalaie, A. Heydarnoori, P. Jamshidi, Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture, IEEE Softw. 33 (2016) 42–52. https://doi.org/10.1109/MS.2016.64.

[39] C. Sürücü, B. Song, J. Krüger, G. Saake, T. Leich, Establishing Key Performance Indicators for Measuring Software-Development Processes at a Large Organization, in: Proc. 28th ACM Jt. Meet. Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng., Association for Computing Machinery, New York, NY, USA, 2020: pp. 1331–1341. https://doi.org/10.1145/3368089.3417057.

[40] V. Ostakhov, N. Artykulna, V. Morozov, Models of IT Projects KPIs and Metrics, in: 2018 IEEE Second Int. Conf. Data Stream Min. Process., 2018: pp. 50–55. https://doi.org/10.1109/DSMP.2018.8478464.

[41] L. Lopez, A. Bagnato, A. Ahberve, X. Franch, QFL: Data-Driven Feedback Loop to Manage Quality in Agile Development, in: 2021 IEEE/ACM 43rd Int. Conf. Softw. Eng. Softw. Eng. Soc., 2021: pp. 58–66. https://doi.org/10.1109/ICSE-SEIS52602.2021.00015.

[42] M.A. López-Peña, J. Díaz, J.E. Pérez, H. Humanes, DevOps for IoT Systems: Fast and Continuous Monitoring Feedback of System Availability, IEEE Internet Things J. 7 (2020) 10695–10707. https://doi.org/10.1109/JIOT.2020.3012763.

[43] A.R. Munappy, D.I. Mattos, J. Bosch, H.H. Olsson, A. Dakkak, From Ad-Hoc Data Analytics to DataOps, in: Proc. Int. Conf. Softw. Syst. Process., Association for Computing Machinery, New York, NY, USA, 2020: pp. 165–174. https://doi.org/10.1145/3379177.3388909.

[44] P. YELLAND, A Modern Retail Forecasting System in Production., Foresight Int. J. Appl. Forecast. (2020) 5–15. https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=146375764&lang=pt-pt&site=ehost-live&scope=site.

[45] Y. Bobbert, J. Scheerder, On the Design and Engineering of a Zero Trust Security Artefact, Adv. Intell. Syst. Comput. 1363 AISC (2021) 830–848. https://doi.org/10.1007/978-3-030-73100-7_58.

[46] C. Marnewick, J. Langerman, DevOps and Organizational Performance: The Fallacy of

Chasing Maturity, IEEE Softw. 38 (2021) 48–55. https://doi.org/10.1109/MS.2020.3023298.

[47] A. Janes, V. Lenarduzzi, A.C. Stan, A Continuous Software Quality Monitoring Approach for Small and Medium Enterprises, in: Proc. 8th ACM/SPEC Int. Conf. Perform. Eng. Companion, Association for Computing Machinery, New York, NY, USA, 2017: pp. 97–100. https://doi.org/10.1145/3053600.3053618.

[48] R.E. Raygan, S. Henry, Manifesto for Enterprise Agility, in: 2019 Int. Symp. Syst. Eng., 2019: pp. 1–6. https://doi.org/10.1109/ISSE46696.2019.8984448.

[49] M.A. McCarthy, L.M. Herger, S.M. Khan, B.M. Belgodere, Composable DevOps: Automated Ontology Based DevOps Maturity Analysis, in: 2015 IEEE Int. Conf. Serv. Comput., 2015: pp. 600–607. https://doi.org/10.1109/SCC.2015.87.

[50] O.E. Adalı, Ö. Özcan-Top, O. Demirörs, Evaluation of agility assessment tools: A multiple case study, Commun. Comput. Inf. Sci. 609 (2016) 135–149. https://doi.org/10.1007/978-3-319-38980-6_11.

[51] R. Mahdavi-Hezaveh, J. Dremann, L. Williams, Software development with feature toggles: practices used by practitioners, Empir. Softw. Eng. 26 (2021). https://doi.org/10.1007/s10664-020-09901-z.

[52] N. Rathod, A. Surve, Test orchestration a framework for Continuous Integration and Continuous deployment, in: 2015 Int. Conf. Pervasive Comput., 2015: pp. 1–5. https://doi.org/10.1109/PERVASIVE.2015.7087120.

[53] L. Lopez, M. Manzano, C. Gomez, M. Oriol, C. Farre, X. Franch, S. Martinez-Fernandez, A.M. Vollmer, QaSD: A Quality-aware Strategic Dashboard for supporting decision makers in Agile Software Development, Sci. Comput. Program. 202 (2021). https://doi.org/10.1016/j.scico.2020.102568.

[54] H. Huijgens, D. Spadini, D. Stevens, N. Visser, A. van Deursen, Software Analytics in Continuous Delivery: A Case Study on Success Factors, in: Proc. 12th ACM/IEEE Int. Symp. Empir. Softw. Eng. Meas., Association for Computing Machinery, New York, NY, USA, 2018. https://doi.org/10.1145/3239235.3240505.

[55] S. Neely, S. Stolt, Continuous Delivery? Easy! Just Change Everything (Well, Maybe It Is Not That Easy), in: 2013 Agil. Conf., 2013: pp. 121–128. https://doi.org/10.1109/AGILE.2013.17.

[56] C.C. Harris, AGILE SOFTWARE DEVELOPMENT: DHS Has Made Significant Progress in Implementing Leading Practices, but Needs to Take Additional Actions., GAO Reports. (2020) i–107. https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=143568729&lang=pt-pt&site=ehost-live&scope=site.

[57] A.R.T. de Lacerda, C.S.R. Aguiar, FLOSS FAQ Chatbot Project Reuse: How to Allow Nonexperts to Develop a Chatbot, in: Proc. 15th Int. Symp. Open Collab., Association for Computing Machinery, New York, NY, USA, 2019. https://doi.org/10.1145/3306446.3340823.

[58] W. Mohsen, M. Aref, K. ElBahnasy, Scaled Scrum Framework for Cooperative Domain Ontology Evolution, in: Proc. 2020 6th Int. Conf. Front. Educ. Technol., Association for Computing Machinery, New York, NY, USA, 2020: pp. 135–143. https://doi.org/10.1145/3404709.3404770.

[59] P.P. Than, M.P. Phyu, Continuous Integration for Laravel Applications with GitLab, in: Proc. Int. Conf. Adv. Inf. Sci. Syst., Association for Computing Machinery, New York, NY, USA, 2019. https://doi.org/10.1145/3373477.3373479.

[60] N. Forsgren, M.-A. Storey, C. Maddila, T. Zimmermann, B. Houck, J. Butler, The SPACE of Developer Productivity: There's More to It than You Think., Queue. 19 (2021) 20–48. https://doi.org/10.1145/3454122.3454124.

[61] C. Werner, Z.S. Li, D. Lowlind, O. Elazhary, N.A. Ernst, D. Damian, Continuously Managing NFRs: Opportunities and Challenges in Practice, IEEE Trans. Softw. Eng. (2021) 1. https://doi.org/10.1109/TSE.2021.3066330.

[62] Y. Wang, M. Mäntylä, S. Eldh, J. Markkula, K. Wiklund, T. Kairi, P. Raulamo-Jurvanen, A. Haukinen, A Self-Assessment Instrument for Assessing Test Automation Maturity, in: Proc. Eval. Assess. Softw. Eng., Association for Computing Machinery, New York, NY, USA, 2019: pp. 145–154. https://doi.org/10.1145/3319008.3319020.

[63] Y. Wang, M. Pyhäjärvi, M. V Mäntylä, Test Automation Process Improvement in a DevOps Team: Experience Report, in: 2020 IEEE Int. Conf. Softw. Testing, Verif. Valid. Work., 2020: pp. 314–321. https://doi.org/10.1109/ICSTW50294.2020.00057.

[64] A. Leinonen, V. Roto, Service Design Handover to user experience design – a systematic literature review, Inf. Softw. Technol. 154 (2023) 107087.

https://doi.org/https://doi.org/10.1016/j.infsof.2022.107087.

[65] H. Huijgens, A. van Deursen, R. van Solingen, Success Factors in Managing Legacy System Evolution: A Case Study, in: 2016 IEEE/ACM Int. Conf. Softw. Syst. Process., 2016: pp. 96–105. https://doi.org/10.1109/ICSSP.2016.021.

[66] B. Snyder, B. Curtis, Using Analytics to Guide Improvement during an Agile–DevOps Transformation, IEEE Softw. 35 (2018) 78–83. https://doi.org/10.1109/MS.2017.4541032.

[67] G. Kulkarni, Transitioning an Enterprise From COBIT 5 to COBIT 2019., COBIT Focus. (2019) 1–9. https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=139604912&lang=pt-pt&site=ehost-live&scope=site.

[68] C. Margineanu, C. Grigoras, M. Carabas, S. Weisz, D. Mihai, M.-E. Mihailescu, N. Tapus, Client Request Analysis Tool for CERN ALICE Grid Services, in: 2020 Int. Conf. Comput. Data Sci., 2020: pp. 462–467. https://doi.org/10.1109/CDS49703.2020.00097.

[69] J.S. Saltz, I. Shamshurin, Achieving Agile Big Data Science: The Evolution of a Team's Agile Process Methodology, in: 2019 IEEE Int. Conf. Big Data (Big Data), 2019: pp. 3477–3485. https://doi.org/10.1109/BigData47090.2019.9005493.

[70] D.M. Szabó, J.-P. Steghöfer, Coping Strategies for Temporal, Geographical and Sociocultural Distances in Agile GSD: A Case Study, in: 2019 IEEE/ACM 41st Int. Conf. Softw. Eng. Softw. Eng. Pract., 2019: pp. 161–170. https://doi.org/10.1109/ICSE-SEIP.2019.00025.

[71] R.K. Gupta, M. Venkatachalapathy, F.K. Jeberla, Challenges in Adopting Continuous Delivery and DevOps in a Globally Distributed Product Team: A Case Study of a Healthcare Organization, in: 2019 ACM/IEEE 14th Int. Conf. Glob. Softw. Eng., 2019: pp. 30–34. https://doi.org/10.1109/ICGSE.2019.00020.

[72] U. Telemaco, T. Oliveira, P. Alencar, D. Cowan, A Catalogue of Agile Smells for Agility Assessment, IEEE Access. 8 (2020) 79239–79259. https://doi.org/10.1109/ACCESS.2020.2989106.

[73] H. Huijgens, R. Lamping, D. Stevens, H. Rothengatter, G. Gousios, D. Romano, Strong Agile Metrics: Mining Log Data to Determine Predictive Power of Software Metrics for Continuous Delivery Teams, in: Proc. 2017 11th Jt. Meet. Found. Softw. Eng., Association for Computing Machinery, New York, NY, USA, 2017: pp. 866–871.

https://doi.org/10.1145/3106237.3117779.

[74] K. Pedretti, A.J. Younge, S.D. Hammond, J.H.L. III, M.L. Curry, M.J. Aguilar, R.J. Hoekstra, R. Brightwell, Chronicles of Astra: Challenges and Lessons from the First Petascale Arm Supercomputer, in: SC20 Int. Conf. High Perform. Comput. Networking, Storage Anal., 2020: pp. 1–14. https://doi.org/10.1109/SC41405.2020.00052.

[75] T.A. Limoncelli, The Small Batches Principle, Commun. ACM. 59 (2016) 52–57. https://doi.org/10.1145/2909468.

[76] N. Kerzazi, B. Adams, Botched Releases: Do We Need to Roll Back? Empirical Study on a Commercial Web App, in: 2016 IEEE 23rd Int. Conf. Softw. Anal. Evol. Reengineering, 2016: pp. 574–583. https://doi.org/10.1109/SANER.2016.114.

[77] B.S. Farroha, D.L. Farroha, A Framework for Managing Mission Needs, Compliance, and Trust in the DevOps Environment, in: 2014 IEEE Mil. Commun. Conf., 2014: pp. 288–293. https://doi.org/10.1109/MILCOM.2014.54.

[78] S.M. Syed-Mohamad, N.S.M. Akhir, SoReady: An Extension of the Test and Defect Coverage-Based Analytics Model for Pull-Based Software Development, in: 2019 26th Asia-Pacific Softw. Eng. Conf., 2019: pp. 9–14. https://doi.org/10.1109/APSEC48747.2019.00011.

[79] J. Ludwig, D. Cline, A. Novstrup, A Case Study Using CBR-Insight to Visualize Source Code Quality, in: 2020 IEEE Aerosp. Conf., 2020: pp. 1–12. https://doi.org/10.1109/AERO47225.2020.9172526.

[80] C. Bansal, S. Renganathan, A. Asudani, O. Midy, M. Janakiraman, DeCaf: Diagnosing and Triaging Performance Issues in Large-Scale Cloud Services, in: 2020 IEEE/ACM 42nd Int. Conf. Softw. Eng. Softw. Eng. Pract., 2020: pp. 201–210.

[81] R. Fidel, The case study method: A case study, Libr. Inf. Sci. Res. 6 (1984) 273–288.

[82] J. Faustino, R. Pereira, B. Alturas, M.M. da Silva, Agile information technology service management with DevOps: An incident management case study, Int. J. Agil. Syst. Manag. 13 (2020) 339 – 389. https://doi.org/10.1504/IJASM.2020.112331.

[83] P.R. Newswire, Leading Medical Device Companies Leverage Parasoft to Continuously Evolve Embedded Software Development Workflow, PARASOFT-Partners. (2021). https://search.ebscohost.com/login.aspx?direct=true&db=bwh&AN=202107201241PR.N

EWS.USPR.LA49262&lang=pt-pt&site=ehost-live&scope=site.

[84] K. Bernsmed, M.G. Jaatun, Threat modelling and agile software development: Identified practice in four Norwegian organisations, in: 2019 Int. Conf. Cyber Secur. Prot. Digit. Serv. (Cyber Secur., 2019: pp. 1–8. https://doi.org/10.1109/CyberSecPODS.2019.8885144.

[85] L.A. Rosser, J.H. Norton, A Systems Perspective on Technical Debt, in: 2021 IEEE Aerosp. Conf., 2021: pp. 1–10. https://doi.org/10.1109/AERO50100.2021.9438359.

[86] N. Tomas, J. Li, H. Huang, An Empirical Study on Culture, Automation, Measurement, and Sharing of DevSecOps, in: 2019 Int. Conf. Cyber Secur. Prot. Digit. Serv. (Cyber Secur., 2019: pp. 1–8. https://doi.org/10.1109/CyberSecPODS.2019.8884935.

7 ANNEXES

Annex 1- Agile Practices

Practices	Articles
Continuous delivery	[1,9,15,38,42,48,51–56,66]
Continuous integration	[1,2,7,33,37,51,52,56–59,66]
Continuous deployment	[1,3,7,33,52,55,57,58,60]
Continuous feedback	[1,6,7,33,41,42,61]
Continuous test and automation	[1,15,33,36,37,62,83]
Continuous improvement mindset	[1,7,42,49]
Continuous learning/experimentation/monitoring	[1,7,38,42]
Meeting daily to ease the communication of team members	[7,30,56]
Prioritizing requirements in a backlog based on value	[7,56,65]
Continuous collaboration	[1,42]
Continuous planning	[7,56]
Ensuring the quality of code being developed	[7,56]
Maintain a sustainable development pace	[7,56]
Self-organizing Agile teams	[7,56]
Acquisition policy and procedures support Agile culture	[56]
Align clearly goals and objectives	[56]
Aligning incentives and rewards to Agile culture	[56]
Cascading sponsorship for agile software development	[56]
Continuous vulnerability assessment	[33]
Creating user stories to define work	[56]
Define and incorporate non-functional requirements	[56]
Defining the role of a product owner	[56]
Establish appropriate life cycle activities	[56]
Estimating the relative complexity of user stories	[56]
Involving the developers	[84]
Make technical and project support tools available	[56]
Observing end-iteration demonstrations	[56]
Observing end-iteration retrospectives	[56]
Self-assessment	[33]
Sponsorship understanding for agile software development	[56]
Stakeholder Participation	[33]
Threat modelling	[84]
Training all program staff	[56]
Triggering the threat modelling activities	[84]
Using checklists and clearly defined processes and routines	[84]

Annex 1 Factors that enable Agile Adoption

<table>
<tr><th colspan="2"></th><th>Factor</th><th>Articles</th></tr>
<tr><td rowspan="15">Enabler</td><td rowspan="8">Cultural Enabler</td><td>Collective Ownership Shared Values</td><td rowspan="15">[9,46,49]</td></tr>
<tr><td>Definition of Success</td></tr>
<tr><td>Effortless Communication Continuous Experimentation and Learning</td></tr>
<tr><td>Incentives</td></tr>
<tr><td>Respect and Trust Constant</td></tr>
<tr><td>Responsibility</td></tr>
<tr><td>Shared Goals and Mindset</td></tr>
<tr><td>Shared Ways of Working</td></tr>
<tr><td rowspan="7">Technological Enabler</td><td>Build Automation</td></tr>
<tr><td>Configuration Management</td></tr>
<tr><td>Deployment Automation</td></tr>
<tr><td>Infrastructure Automation</td></tr>
<tr><td>Monitoring Automation</td></tr>
<tr><td>Recovery Automation</td></tr>
<tr><td>Test Management</td></tr>
</table>

Annex 2 Factors that disable Agile Adoption

	Factor	Articles
Impediments	Communications and bureaucratic barriers	[1,3,15,24,30,69]
	Geographical Distribution	[30,46,70,71]
	Lack of Education Around DevOps	[3,46,71,72]
	Cultural Changes	[3,28,70]
	Processes differ country to country	[69–71]
	Absence of Frequent Interactions and Deliveries	[71,72]
	Absence of Test-driven Development	[72,85]
	Complex Tasks	[72,85]
	Dependence on Internal Specialists	[72,85]
	DevOps Requires Both Dev and Ops Skills and Knowledge	[46,71]
	Difficulty to define goals	[46,72]
	Lack of motivation	[46,84]
	Lack of proper tools and knowledge about how to use those	[66,86]
	Unplanned Work/ lack of backlog prioritization	[71,72]
	Absence of Timeboxed Iteration and meeting	[72]
	Communicate what kinds of results are expected	[11]
	DevOps Is More Work for Developers	[46]
	Identifying relevant threats	[84]
	Iteration Start without an Estimated Effort	[72]
	No Iteration Retrospective	[72]
	Iterations with Different Duration	[72]
	Knowing the “definition of done”	[84]
	Lack of continuous requirements	[71]
	Lack of Strategic Direction from Senior Management	[46]
	Large Development Team	[72]
	Long Break Between Iterations	[72]
	Measure if transformation has reached its goals	[11]
	Monolithic Architecture	[46]
	Multiple Production Environments	[46]
	Resistance to change	[46]
	Security measurements are rarely applied	[86]
	Shared Developers	[72]
	Threat modelling is time-consuming	[84]
	Unfinished Work in a Closed Iteration	[72]

Annex 3 Benefits of adopt Agile Metodologies

Benefits	articles
Faster feedback	[1,6,7,33,41,42,61,73]
Collaborative work	[13,24,42,43,74]
Faster fixes	[31,43,51,75]
Reduced technological and human risk	[9,14,45,75]
Multidisciplinary and self-improving teams	[13,30,43]
Features arrived faster	[9,60,75]
Rapid change	[14,42,51]
Created a culture of optimization and automation	[5,75]
Improve ability to innovate and productivity	[9,75]
Less waste (less code redone if errors are found earlier)	[9,75]
More motivated, happier, and gratified and less stressed team	[24,75]
Push to production small batches is easier	[51,75]
Reduce of Latency	[24,75]
Speeding up software development	[3,14]
Agility	[43]
Better performance	[46]
Company more competitive and confident	[75]
Happier customers	[75]
Improved process of documentation and automation	[75]
Increased team flexibility	[43]

Annex 4 - Results for Sprint 1

Type	KPI	
Delivery	KPI1	-
	KPI2	-
	KPI3	-
	KPI4	STRY1 - 30 days STRY2 - 11 days STRY3 - 34 days STRY4 - 30 days STRY5 - 5 days STRY6 - 30 days STRY7 - 6 days STRY8 - 36 days STRY9 - 34 days
Costs	KPI5	-
	KPI6	-
Defects	KPI7	3/8 = 0,38
	KPI8	-
	KPI9	0
	KPI10	2/8 = 0,25
	KPI11	Priority 1= 2 Priority 3= 2 Priority 4= 2
Tests	KPI12	100,00
	KPI13	-

Annex 5 - Results for Sprint 2

Type	**KPI**	
Delivery	KPI1	-
	KPI2	-
	KPI3	-
	KPI4	STRY1- 15 days STRY2- 100 days STRY3- 21 days STRY4- 20 days STRY5- 6 days STRY6- 19 days STRY7- 22 days STRY8- 1 days STRY9- 19 days STRY10- 19 days STRY11- 95 days
Costs	KPI5	-
	KPI6	-
Defects	KPI7	5/8 = 0,63
	KPI8	-
	KPI9	0
	KPI10	2/8 = 0,25
	KPI11	Priority 1= 2 Priority 2= 2 Priority 3= 1 Priority 4= 1 Priority 5= 1
Tests	KPI12	(32*100) / 34 = 94,12
	KPI13	-

Annex 6 - Results for Sprint 3

Type	KPI	
Delivery	KPI1	-
	KPI2	-
	KPI3	-
	KPI4	STRY1- 3 days STRY2- 4 days STRY3- 1 days STRY4- 45 days STRY5- 5 days STRY6- 3 days STRY7- 3 days STRY8- 6 days STRY9- 5 days STRY10- 50 days STRY11- 51 days STRY12- 57 days STRY13- 56 days
Costs	KPI5	-
	KPI6	-
Defects	KPI7	2/8 = 0,25
	KPI8	-
	KPI9	0
	KPI10	0/8 = 0
	KPI11	Priority 3= 2
Tests	KPI12	(27*100) / 28 = 96,43
	KPI13	-