Adopting DevOps: A Grounded Theory of Well Succeeded DevOps Adoption in Practice

Michael Shell
School of Electrical and
Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332–0250
Email: http://www.michaelshell.org/contact.html

Homer Simpson
Twentieth Century Fox
Springfield, USA
Email: homer@thesimpsons.com

James Kirk
and Montgomery Scott
Starfleet Academy
San Francisco, California 96678–2391
Telephone: (800) 555–1212
Fax: (888) 555–1212

***Abstract*—DevOps is a term that has emerged in the industry. Using predominantly literature reviews, academic studies have sought to characterize it as a set of concepts with related practices. This view of DevOps does not provides clear guidance to companies that want to adopt DevOps, changing your software development paradigm. Empirical studies on industry, where the term DevOps was born, seem to be an appropriate approach to investigate DevOps adoption with more clear guidance to new practitioners. Based on a Grounded Theory study of 15 well succeeded DevOps adoptions in companies across five countries, we present a model to improve the understanding and guidance of DevOps adoption. The model increments the existing view of DevOps providing explanation about the role and motivation of each element in a DevOps adoption process and the relationship between them. The model was applied in the DevOps adoption of Brazilian Federal Court of Accounts, a company of Brazilian Federal Government.**

I. Introduction

DevOps is a term that has emerged in software development industry. Even before the existence of the term - a mix of ”development” and ”operations” words, that has been used since 2009 - companies like Flickr [3] already pointed out the need to break the existent paradigms about the role of operations in software development teams. Since then, the term that has appeared without clear delimitation has gained strength and interest in companies that can perceive the benefits of apply agile practices in operation tasks too. However, adopt DevOps still as a challenge task.

Researchers have proposed a number of characterizations of DevOps as a set of concepts with related practices, and using predominantly literature reviews [8], [16], [14], [15], [6] and [9]. Despite of some these studies include some interviews with industry practitioners, the interviews are used merely to complement the literature review data, not having a leading role. Consequently, the obtained DevOps characterizations allow a comprehensive understanding of the elements that constitute it, but they don't provide detailed guidance to new candidates that want to adopt DevOps. This implies several open questions: Is there a suitable way to adopt DevOps? Do the different elements of DevOps have the same weight in a DevOps Adoption? For example, according to the framework proposed in [14] automation is one of four dimensions. So, if, by example, one team automated all of your procedures, will be him reached 25% of DevOps adoption? Where, why and how elements like measurement, sharing, automation and others appear in a DevOps adoption? To answer these questions, it is necessary a holistic understanding of how software development teams that have successfully adopted DevOps did this.

In this paper, we present a model of how DevOps was successfully adopted in 15 companies across five countries. The model was constructed based on a Grounded Theory study and was applied in an important Brazilian federal government institution. We find the elements that make up the adoption of DevOps and most of them are similar to those present in related work. Considering that we start from a limited exposure to existing literature, due the nature of a grounded theory study, this study validates the previous research in the parts where they coincide.

We find that DevOps adoption involves a very specific relationship between eight categories: agility, automation, collaboration culture, continuous measurement, quality assurance, resilience, sharing and transparency. The core category and validation point of DevOps adoption is collaboration culture. One part of the identified categories (automation, sharing and transparency) exists as means typically used to propitiate the formation of a collaboration culture. Another part of categories are expected or required consequences of this formation (agility and resilience). And, other two categories (continuous measurement and quality assurance) can work in the two ways mentioned above.

The main contributions of this paper are: improve the guidance about how do adopt DevOps based on an industry view; provide one detailed example of Grounded Theory use in a DevOps investigation, which had not been done before.

The rest of the paper is structured as follows: section II summarizes the related work. In section III, the application of Grounded Theory in this study is described with examples. Section IV contains the results with detailed description of how to adopt DevOps. Section V presents details about categories and related concepts; In section VI is presented a case study that demonstrates the potential application of the results. The paper concludes in section VII.

II. Related Work

A literature review of 2014 [8] proposed eight main concepts related to DevOps: culture, automation, measurement, sharing, services, quality assurance, structures and standards. The research points out that four of this concepts (culture, automation, measurement and sharing) are related to CAMS framework, proposed in [18], which multiple articles referred to. The paper concludes that there is a great opportunity for empirical researchers to study organizations experimenting with DevOps.

Since then, some studies have mixed literature reviews with empirical investigations in the industry [16], [9] and [14].

In [14], is presented a framework with four main dimensions that characterize DevOps: collaboration, automation, measurement and monitoring. The study advocates to confirm three elements of DevOps presented in previous research and to add a new element (monitoring). The paper does not indicates the role of each dimension in a DevOps adoption, causing some gaps: DevOps only make sense with all dimensions being improved at the same time? Does evolving into any single dimension provide proper adoption way of DevOps? If automation, by example, is fully attended, is the DevOps adoption in 25% of progress? Depite of theses gaps, the paper offers an abrangent view of DevOps and the results combine literature review with a short direct investigation of industry through interviews with three practitioners. Except monitoring that was considered part of continuous measurement, the another dimensions are present too in our results, which validates them. The paper also concludes that still a need for empirical research that investigates DevOps adoption in order to validate and enhance the presented conceptual framework.

One extension of [14] was proposed in [15]. This extension added culture as a new dimension of DevOps characterization and presented practices related to each dimension. The data was collected using multivocal literature review combined with three semi-structured interviews. The data analysis was performed using coding techniques similar to those used in Grounded Theory studies. In our study, culture is grouped with collaboration in one category named c̈ollaboration culture,̨ so this five dimensions are presented in our results as only three, but with similar descriptions. The paper present as future work the need to obtain empirical evidence of DevOps adoption in companies that claim to have implemented it, exactly as we are proposing here.

Another source of useful elements that can help in a DevOps adoption is presented in [6]. Applying some Grounded Theory techniques to data analysis in data collected from multiple sources, including gray literature, some important results are presented: a definition to DevOps, issues motivating its adoption, driving principles, the potential benefits and challenges of adopting DevOps and 51 suggested practices and the required skills.

Starting from the same premise that there is an increasing need for software organizations to understand how to adopt DevOps successfully, in [16] was performed a new literature review combined with interview of 13 subjects of one company adopting DevOps. The main results are a set of capabilities, enablers and impediments of DevOps adoption. The grouping of elements into enablers is a very similar way to the presented by us to describe the results of DevOps adoption. Our model presents as difference of this the existence of the collaboration culture category as the point of integration and many of elements presented in the study are grouped in categories in our study. By example, there are many enablers presented (build automation, test automation, deployment automation, monitoring automation, recovery automation and infrastructure automation) that appeared as concepts in our study and consequently are grouped into ”automation” category.

One investigation about the ways in which organizations implement DevOps was published in [9]. The study, similarly to the others cited above, combined literature review with some interviews with practitioners. In the literature review part, the papers were labeled and the similar labels are grouped. The 7 top labels are then presented as elements of DevOps usage in literature: culture of collaboration, automation, measurement, sharing, services, quality assurance and governance. Comparing these labels with our categories results, it is possible to note similarities between them. Except services and governance, the other labels appear as categories here. Specifically about services, in which the paper mentions microservices architecture, our model group the granularity of services as a quality assurance aspect of DevOps, and the autonomy of services as a automation aspect. In interviews the paper presented each experience in an isolated way, not seeking to identify common points of understanding to present a unified view of their DevOps experience. The study concludes that further research including several organizations and encompassing later stages of DevOps adoption still necessary to improve the understanding of DevOps adoption in practice.

DevOps is a term that was born in industry. Although the academic studies seek reflect the industry practice, there is a lack of studies investigating it directly. None of the papers cited above focused on directly investigate DevOps adoption in companies that advocates adopt it, and most of them claimed for new research of this type. The presented work has a number of useful elements, but that need to be complemented with a view of how DevOps has been adopted, containing guidance about how to connect all these isolated parts and then enable new candidates to adopt DevOps in a more consistent way.

III. Research Method

We adopted Grounded Theory (GT) as research method. GT is a method originally proposed in [10] which has as distinguishing features the absence of clear research hypothesis upfront and limitation of literature exposure at the beginning of the research. Is a theory-developing approach in contrast with the more traditional theory-testing approach [5].

Some of related work claimed to used ”GT inspired” approaches. It is a very common rhetoric in recent research on software engineering, but only research that embodies GTs core principles should claim to be a grounded theory study [17].

GT was employed as the research method for the following reasons:

- GT is a consolidated method in other areas of research like sociology, nursing, education and finances and is increasingly being employed to study software engineering topics [17];
- GT is considered an adequate approach to investigate scenarios with questions such as *what's going on here?* [4], which is exactly the scenario proposed here: what's going on DevOps adoption?
- GT allows the researcher to get descriptions without bias of previous researches, which is adequate to collect empirical evidence directly from the practice on industry. The evidence is only reintegrated back with literature after the theory construction.

Since the publication of the original version in [10], several modifications and variations occurred in the original text, coming to exist at least seven different versions of the method [7], where the main versions are those of Glaser or classic, Strauss and Charmaz [17]. The study presented in [17] explores the main aspects of GT versions and establishes that GT studies has to specify which version is used on it. In our study, we choose the classic version. The first reason to our choice is that we did not have a research question at the beginning of the research, exactly as suggested in the classic version. We start from a interest area that are: successfully DevOps adoption in industry. And the second reason is because the more detailed papers in software engineering research use predominantly this version [17].

A. GT Procedures

In Fig. 1, reproduced from [1], is showed the view of the grounded theory procedures that we followed in the conduction of this research.

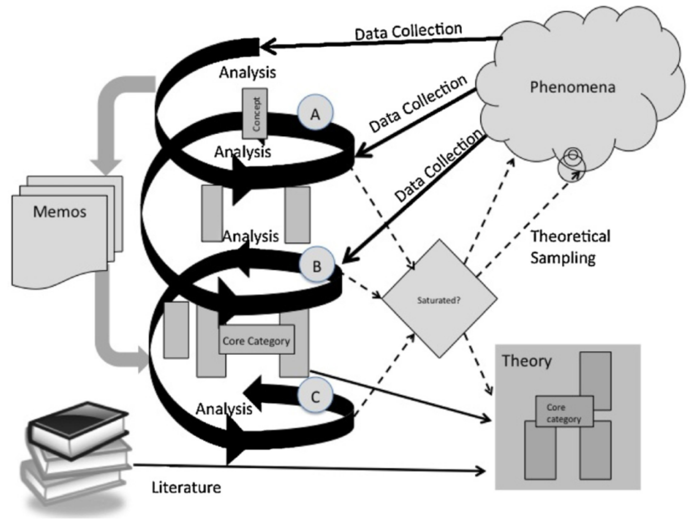

Fig. 1. The GT Method [1]

(A) Initially, we begin collecting data about the adoption process from companies that have successfully adopted it. As the data were collected, they were also analyzed simultaneously. The raw data is analyzed by searching for patterns of incidents to indicate concepts, and concepts grouped into categories. This first step, where all raw data is analyzed, is called open coding [17].

(B) The categories are developed by constant comparison of new incidents with previous. Every grounded theory study has to identify a ”core category” [17]. The core category is responsible for enabling the integration of the other categories and structuring the results into a dense and consolidated grounded theory [13]. The identification of the core category represents the end of open coding and the beginning of the selective coding. In selective coding, only specific variables that are directly related to the core category and their relationships are coded, in order to enable the production of a harmonic theory [5] [12]. Selective coding ends when theoretical saturation is achieved, which occurs when new data (............). Theoretical coding....

(C) After saturation, the resulting theory is reintegrated back into literature comparing with existing theories. The literature search is performed only later to avoid forcing preconceived concepts on the theory development [2].

(D) Throughout the process, memos are wrote capturing thoughts and analytic processes; the memos support the emerging concepts, categories, and their relationships [2].

The following sub-sections present details of procedures applied in this study, containing some examples to illustrate their application.

B. Data Collection

We conducted semi-structured interviews with 15 practitioners of companies from Brazil, Portugal, Ireland, United States and Spain. This practitioners claim to have participated to DevOps adoption process in your companies. Participants were recruited through direct contact in DevOps days event and some general calls for participation posted on DevOps user groups, social networking sites and local communities. A variety of company types were covered in order to achieve a heterogeneous perspective, increasing the potential of generalization of the results. I presents the participant characteristics.

The first column refers to participant numbers P1-P15 to maintain anonymity in conformance with the human ethics guidelines. The second column the role of each participant in your respective company. The remain columns list their years of total experience in software development (SWX), years of experience with DevOps (DX), country of work (CN), main domain of the company, and company sizes (CS).

The interview were conducted over one year using Skype calls and lasted approximately 30 minutes on average.

Data collection and analysis were iterative so the collected data helped guide future interviews. The questions of interview evolved according to the progress of the research...

TABLE I
PARTICIPANT PROFILE

P#	Role	SWX	DX	CN	Domain	CS
P1	DevOps Developer	9	2	IR	IT	S
P2	DevOps Consult.	9	3	BR	IT	M
P3	DevOps Developer	8	1	IR	IT	S
P4	Computer Tech.	10	2	BR	Health	S
P5	Systems Engineer	10	3	SP	Telecom	XL
P6	Developer	3	1	PO	IT	S
P7	Support Analyst	15	2	BR	Telecom	L
P8	DevOps Engineer	20	9	BR	Imob?	M
P9	IT Manager	14	8	BR	IT	M
P10	Network Admin.	15	3	BR	IT	S
P11	Embras	0	0	BR	*	*
P12	Sprinklr	0	0	US	*	*
P13	IFood	0	0	BR	*	*
P14	IT Manager	0	0	BR	*	*
P15	Daniel Almeida	0	0	BR	*	*

C. Data Analysis

The interviews were voice recorded, transcribed, and analyzed. The first moment of the analyze, called open coding in GT, started immediately after transcription of the first interview. So the results of the analyze of the first interview already server to evolve the interview script to the second and so on.

The open coding lasted until there was no doubt about the core category of the study. Similar to the described [2] we have started with a strong candidate to core category that has not been consolidated. Until the analyze of the four interview, we have cogitated automation as core category because it is a recurring pattern in our data. But, we quickly realized that the "automation" category did not explain most of the behaviors or events in our data. At the same time we start to perceive that the collaboration culture also appeared recurrently in the analysis and with more potential to explain the remaining events. We then started to ask explicitly about the role of the automation and how to the collaboration culture is formed in a DevOps adoption process.

After the adaptations made in the script and analysis of new data in a constant comparison process, taking into account the previous analyses and the respective memos wrote during all the process, in the analysis of the tenth interview, we concluded that was unequivocal the core role of the category "collaboration culture" in the explanation about how DevOps was successfully adopted in the companies. At this moment, the open coded ended and the selective coding started.

After the discovery of the core category, we start to restrict the coding only to specific variables that are directly related to the core category and their relationships: the selective coding.

With more three interviews and respective analyze, we started to realize that the new data added less and less content to the emerging theory. The explanation around how the collaboration culture is developed providing DevOps adoption showed signs of saturation. We then conducted more two interviews to conclude that we had reached the theoretical saturation.

After saturation, we start the theoretical coding to find a way to integrate all the concepts, categories and memos in the form of a cohesive and homogeneous theory, where we have pointed out the role of the categories as enablers and outcomes, as shown above.

To illustrate the coding procedures, we present an example of working from interview transcript to the findings for one of the categories: automation.

Apresentar aqui a exemplificacao da codificacao....

D. Reintegrate with Literature (?)

IV. RESULTS

The results of a grounded theory study, as the name of the method itself suggests, are grounded on the collected data, so the hypotheses emerged from data. A grounded theory should describe the key relationships between those categories, i.e. a set of inter-related hypotheses [11]. We present the main categories of the grounded theory of DevOps adoption as a network of three categories of enablers (automation, sharing and transparency) that are commonly used to develop the core category "collaboration culture". This adoption process typically produces concepts related to two categories of outcomes: agility and resilience. And there are two categories that can be considered both as enablers and as outcomes: continuous measurement and quality assurance. In this section we describe the relationships between those categories, i.e. the hypotheses.

A. How to Adopt DevOps?

At the introduction of the paper, we have made the question: it there a suitable way to adopt DevOps? Here, we present one possible response, based on the analyses performed as detailed in the previous section. The main point which should be sought is the formation of a **collaboration culture** between the teams and activities of operations related to software development. The other categories, many of which are also present in other studies that have investigated DevOps, only make sense if the practices and concepts related to them generate some increase in the level of collaboration culture or if this increase produces expected or necessary consequences on it.

H1: *There is a group of categories related to DevOps adoption that only make senses if are used to increase the collaboration culture level. We call this group of categories of **enablers***. If all is automated: deployment, infrastructure providing, monitoring, etc, but this automation is maintained into a silo, where only one people or one team is able or responsible to understand, adapt or evolve this automation, there is no increase in collaboration level and, consequently, the DevOps adoption has not advanced. The same is valid to the other categories of enablers. If transparency, sharing, etc don't contribute to the collaboration culture, they don't contribute to DevOps adoption.

"This midfield between dev and ops was missing, and this tools that we used were just to connect the DevOps" Fire P3

H2: *There is a group of categories related to DevOps adoption that don't contributes to increase of collaboration culture level, but that are pointed out as DevOps adoption related, because they emerge as expected or necessary consequence of* *the adoption. We call this group of categories of* ***outcomes***. In a first moment, the simple fact that a team is more agile in delivering software, or more resilient in failure recovery, don't contributes directly to bring operations tasks closer to development tasks. But, the respondents of the interviews frequently cited the capacity of continuously delivery software and the strongly resilient infrastructure as part of their DevOps adoption process.

H3: *The categories* ***Continuous Measurement*** *and* ***Quality Assurance*** *are related both to DevOps enabling capacity and to DevOps outcoming*. Measurement is cited as a typical responsibility of the operations team. At the same time that the sharing of this responsibility contributes to reduce the silo, it is too cited as a necessary consequence after DevOps adoption, because the context of agility with continuous delivery of software requires more caution, which is supplied by the concepts related to **continuous measurement** category. The same premise is valid to **quality assurance** category. At first glance **quality assurance** appears as one response to the context of agility in operations provided by DevOps adoption. But, the efforts in quality assurance of software products increase the confidence between the dev and ops teams developing the level of **collaboration culture**.

B. DevOps Enablers

DevOps enablers are exactly the means commonly used to increase the level of the collaboration culture in a DevOps adoption process. We have identified five categories of DevOps enablers:

- Automation;
- Continuous Measurement*;
- Quality Assurance*;
- Sharing; and
- Transparency.

* This category can be outcome too

H4: *There is no precedence between enablers in a DevOps adoption process*. We have perceived that the adoption process can follow a path that prioritizes specific enablers, on the condition that the level of collaboration culture increases. And there is no an enabler with evidence that can be more efficient to another in collaboration culture development. For example, in 14 interviews **automation** was cited as a very important enabler to adopt DevOps. But, one respondent answered the following:

"Embora a gente atualmente use automacao em um numero razoavel de cenarios, nos conseguimos desenvolver bastante a nossa cultura sem uso de automacao e eu penso que pode-se adotar DevOps com pouco ou ate mesmo nada de automacao" P8

That is, although automation is a very commonly used enabler, it is possible to increase the level of collaboration culture without focus on automating. And this premise is valid to the other enablers. Each enabler will be detailed in the sequence of the paper.

C. DevOps Outcomes

DevOps outcomes is that group of categories doesn't produces primarily the expected effect of a DevOps enabler, typically concepts that are expected or required consequences of an adoption of DevOps. We have identified four categories that can work as DevOps outcomes:

- Agility;
- Continuous Measurement*;
- Quality Assurance*; and
- Resilience.

* This category can be enabler too

One well succeeded DevOps adoption typically increases the potential of agility, continuous measurement, quality assurance and resilience of a team. But, in some cases this potential is not completely used due business decisions. For example, one respondent has cited that at a first moment the company don't allowed the continuous deployment of applications in production:

"Nos tinhamos condicoes e seguranca para publicar continuamente em ambiente de producao, porem, por os gerentes tinham medo e decidiram por publicar apenas semanalmente em producao" P9

In this case, DevOps increased the level of agility of the company, but, this potential was not totally tapped.

As well as enablers, each outcome will be detailed in the sequence of the paper.

V. Categories and Concepts

In this section we present details and related concepts of each category of DevOps adoption.

A. The Core Category: Collaboration Culture

In this section we present the details of what exactly means build and increase a collaboration culture in DevOps adoption.

The collaboration culture is essentially about remove the barriers that constitutes the silos between development and operations teams or activities. At first glance, it seems somewhat obvious, but the respondents cited some mistakes that they consider recurrent in not prioritize this aspect in a DevOps adoption:

"In a DevOps adoption, there is a very strong cultural issue that the teams sometimes are not adapted. Related to this, one thing that bothers me a lot and that I see happen a lot is people hitching DevOps exclusively to tools" Hepta P9

As before described, in a grounded theory study, the categories emerge from a set of related concepts. The concepts that compose the "collaboration culture" category in our analyses are six:

- Blameless
- Facilitated communication
- Operations in day to day development
- Product thinking
- Shared responsibilities
- Software development empowerment - confidence between teams.

B. Automation

Automation was the category with the higher number of related concepts. This occurs because manual proceedings are considered as strong candidates to propitiate the formation of a silo. If a task is manual, one people, or one team will be responsible to execute it. Although transparency and sharing can be used to ensure collaboration even in manual tasks, with automation the points where silos may arise are minimized.

”When a developer needed to build a new application, the common flow was: he creates a ticket to the operations team, which manually evaluates and creates what was requested. This task could take a lot of time and there was no visibility between teams about what was done. So, here, we adopted this strategy of infrastructure as code where the entire infrastructure is versioned in a common language that anyone, be it the developer, the operations guy or even the manager, he looks and says: the configuration of application x is y, simple”. Sprinklr (P11)

In addition to contribute with transparency, automation is also considered important to ensure the repeatability of the tasks, reducing rework and risk of human failure and, consequently, collaborating to increase the confidence between teams, which is an important aspect of the collaboration culture.

”Before we adopt DevOps, there was a lot of manual work. For example, if you needed to create an database schema, was manual; if you needed to create a database server, was manual; if you needed to create one more EC2 instance, also manually. This manual work was time consuming and often caused errors and rework”. Digitary (P1)

”Our main motivation to adopt DevOps was basically reduce rework. Almost every week, we had to basically make new servers and start it manually, which was very time-consuming” UniOdonto (P4)

We find 8 concepts that were grouped in the automation category:

- Autonomous services: it is one of the characteristics of microservices, frequently cited as part of DevOps adoption. The services need to be independently deployable by fully automated deployment machinery;
- Containerization: docker is cited as a way to automate the provisioning the environment where the application will execute: a container. The container is one frequent way used to automate application execution.
- Deployment automation: use of tools like Docker, Kubernetes, Arch, Spinnaker, AWS CodeDeploy, Microsoft Visual Studio Team Service, etc to automate the deployment task. This automation contributes to repeatability, parity between environments and to reduce risks of human error in the deployment task.
- Infrastructure provisioning automation: use of tools like Terraform, Ansible, Puppet and Chef to provide infrastructure like database and application servers in an automated way.
- Infrastructure management automation: use of tools to management the infrastructure in an automated way. Examples: management of memory and CPU of a host, management of versions of libraries and database versioning.
- Monitoring automation: ability to monitor the application and infrastructure without human intervention. One classic example is the automated sending of alerts (SMS, slack message or even cellphone calls) in case of incidents.
- Recovery automation: ability to, without human intervention, replace one server instance that presents problem or roolback a failed deployment to the previous version.
- Test automation: execute the application tests in an automated way.

C. Continuous Measurement

Continuous measurement has two facets in a DevOps adoption process: it can work as an enabler or as an outcome.

As an enabler, perform regularly the activities of measurement and share this responsibility contributes to avoid this typical silo and reinforce the collaboration culture, because it is a typical responsibility of the operations team.

”Citar trechos de entrevista aqui”

As an outcome, continuously collect metrics from applications and infrastructure is a required consequence of DevOps adoption. It occurs because the resultant agility increases the risk of something going wrong. The team should be able to react quickly in case of problems, and the continuous measurement allows it to be proactive and resilient.

”Citar trechos de entrevista aqui”

- Application log monitoring: every application produces logs, and these logs are an important source of data to be used in the continuous measurement perspective.
- Continuous infrastructure monitoring: the monitoring is not performed by a specific person or team in a specific moment. The responsibility to monitor the infrastructure is shared and it is executed in day by day software development.
- Continuous application measurement: the application is instrumented to provide metrics that are used to evaluate aspects and often direct evolutions or business decisions.
- Monitoring automation: see V-B.

D. Quality Assurance

- Code branching
- Cohesive services: another characteristic of microservices, frequently cited as part of DevOps adoption. The services need to be focused in doing one thing well, which is one quality assurance aspect.
- Continuous testing
- Parity between environments
- Source code static analysis

E. Sharing

- Activities sharing

- Knowledge sharing
- Process sharing

F. Transparency

- Infrastructure as code
- Shared pipelines
- Sharing on a regular basis

G. Agility

- Continuous Integration
- Continuous Delivery
- Continuous deployment: the deployment is a common point of manifestation of the existence of silos, the development team generates one artifact and throw it *over the wall* to be deployed by the operations team.

H. Resilience

- Auto scaling
- Recovery automation
- Zero down-time

VI. Case Study

Our proposed model was applied to guide the DevOps adoption in the Brazilian Federal Court of Accounts (hereafter TCU), one important government institution. The TCU has the evolution of DevOps as one of the strategic objectives of its area of information technology.

Before the application of the model, TCU was produced some result in deployment automation and the focus was being directed to the tooling issue. Although of some advance in the relation between ops and dev teams, this was not placed as the most important point of DevOps adoption.

Acoes desenvolvidas para incrementar a cultura de colaboracao:

- BBT sobre cultura de colaboracao DevOps para disseminar essa informacao;
- Reducao da burocracia na comunicacao entre os times: nada de ticket no servicedesk, canal no slack e aproximacao fisica dos times.
- Treinamentos tipicos de ops extendidos para o time de dev;
- Participacao de ambos os times em eventos de DevOps (devops days, por exemplo);
- Inclusao de DevOps como tema fixo para BBTs;
- Criacao de comite de arquitetura com representacao de todos os times para promover transparencia e padronizacao na execucao de procedimentos

VII. Conclusion

Conclusion

Acknowledgment

The authors would like to thank...

References

[1] Steve Adolph, Wendy Hall, and Philippe Kruchten. Using grounded theory to study the experience of software development. *Empirical Software Engineering*, 16(4):487–513, 2011.

[2] Steve Adolph, Philippe Kruchten, and Wendy Hall. Reconciling perspectives: A grounded theory of how people manage the process of software development. *Journal of Systems and Software*, 85(6):1269–1286, 2012.

[3] John Allspaw and Paul Hammond. 10+ deploys per day: Dev and ops cooperation at flickr. In *Velocity: Web Performance and Operations Conference*, 2009.

[4] Jane H Barnsteiner. Using grounded theory in nursing. *Journal of Advanced Nursing*, 40(3):370–370, 2002.

[5] Gerry Coleman and Rory OConnor. Using grounded theory to understand software process improvement: A study of irish software product companies. *Information and Software Technology*, 49(6):654–667, 2007.

[6] Breno B Nicolau de França, Helvio Jeronimo Junior, and Guilherme Horta Travassos. Characterizing devops by hearing multiple voices. In *Proceedings of the 30th Brazilian Symposium on Software Engineering*, pages 53–62. ACM, 2016.

[7] Norman K Denzin. Grounded theory and the politics of interpretation. *The Sage handbook of grounded theory*, pages 454–471, 2007.

[8] Floris Erich, Chintan Amrit Amrit, and Maia Daneva. Cooperation between information system development and operations: a literature review. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2014*. ACM, 2014.

[9] FMA Erich, C Amrit, and Maya Daneva. A qualitative study of devops usage in practice. *Journal of Software: Evolution and Process*, 29(6), 2017.

[10] BG Glase and Arselm L Strauss. The discovery of grounded theory: Strategies for qualitative research. *New York: Aldlne*, 1967.

[11] Rashina Hoda and James Noble. Becoming agile: a grounded theory of agile transitions in practice. In *Software Engineering (ICSE), 2017 IEEE/ACM 39th International Conference on*, pages 141–151. IEEE, 2017.

[12] Rashina Hoda, James Noble, and Stuart Marshall. The impact of inadequate customer collaboration on self-organizing agile teams. *Information and Software Technology*, 53(5):521–534, 2011.

[13] Sami Jantunen and Donald C Gause. Using a grounded theory approach for exploring software product management challenges. *Journal of Systems and Software*, 95:32–51, 2014.

[14] Lucy Ellen Lwakatare, Pasi Kuvaja, and Markku Oivo. Dimensions of devops. In *International Conference on Agile Software Development*, pages 212–217. Springer, 2015.

[15] Lucy Ellen Lwakatare, Pasi Kuvaja, and Markku Oivo. An exploratory study of devops extending the dimensions of devops with practices. *ICSEA 2016*, page 104, 2016.

[16] Jens Smeds, Kristian Nybom, and Ivan Porres. Devops: a definition and perceived adoption impediments. In *International Conference on Agile Software Development*, pages 166–177. Springer, 2015.

[17] Klaas-Jan Stol, Paul Ralph, and Brian Fitzgerald. Grounded theory in software engineering research: a critical review and guidelines. In *Software Engineering (ICSE), 2016 IEEE/ACM 38th International Conference on*, pages 120–131. IEEE, 2016.

[18] John Willis. What devops means to me. *Chef.io*, 2010. Retrieved from https://blog.chef.io/2010/07/16/what-devops-means-to-me/.