An Empirical Taxonomy of DevOps in Practice

Ruth W. Macarthy
School of Science, Engineering and Environment
University of Salford
Manchester, UK
r.w.macarthy@edu.salford.ac.uk

Julian M. Bass
School of Science, Engineering and Environment
University of Salford
Manchester, UK
j.bass@salford.ac.uk

***Abstract*—DevOps is described as a software engineering culture and philosophy that utilises cross-functional teams to build, test and release software faster and more reliably through automation. Research shows that its adoption speeds up software delivery time, improve quality, security, and collaboration in software development. One controversial issue has been whether DevOps is an organisation-wide culture or a job description. As DevOps is an emerging concept, its definitions and best practices are still hazy, making its implementation in practice less informed and somewhat risky. The rising trend of DevOps adoption among software development practitioners therefore heightens the need for in-depth investigation into its implementation.This paper seeks to contribute to the above by critically examining DevOps implementation in practice through an exploratory case study, based on interviews with 11 industry practitioners across nine organisations. Transcripts of interviews were coded and analysed using a method informed by Grounded Theory. This study provides an empirical taxonomy of DevOps implementation, describing developers' interaction with On-premises Ops, Outsourced Ops, DevOps teams, and DevOps bridge teams. We present a novel mapping of the approaches to on-premises and cloud-based deployments, and identified the facilitators of DevOps practices in the different modes. We further identified three distinct groups of activities in the fourth mode: provisioning and maintenance of physical systems, function virtualisation and creation of automated pipelines, and development, deployment and maintenance of applications, which may have given rise to the implementation of DevOps as bridge teams. Interviewees claimed these distinctions allowed developers to focus on delivering value for the business.**

***Keywords*-DevOps implementation, DevOps practices, Agile operations, Agile deployment**

I. Introduction

The evolution of software development methodologies is in pursuit of optimised coordination of work among the actors in the process, and standardisation [1]. Its continual change is aimed at the faster delivery of quality software. DevOps (Development and Operations) is described as an emerging software engineering culture and philosophy that utilises cross-functional teams (development, operations, security, and QA) to build, test, and release software faster and more reliably through automation [1]–[5]. The strategy promises benefits such as faster delivery, improved quality and security, and better collaboration.

A. The Goal of DevOps

The introduction of DevOps is an attempt to address coordination in software development [1]–[5]. From literature, it is understood that DevOps seeks to bridge the gap between the conflicting priorities of the development and IT Ops teams. According to [6], "DevOps community advocates communication between the operations staff and the development staff as a means of ensuring that the developers understand the issues associated with operations". However, for [7], the actual gaps to be bridged are "disconnects between processes, measurements, technologies, and data". We view the latter as a more granular description of the "gap" between development and deployment teams. [8] argues that three possible approaches to bridging the gap are:

"Mix responsibilities: assign both development and operations responsibilities to all engineers, or Mix personnel: increase communication and collaboration between Dev and Ops, but keep existing roles differentiated, or Bridge team: create a separate DevOps team that functions as a bridge between Devs and Ops".

They carried out a study to investigate the 'mixed responsibilities' approach. DevOps is fast becoming an integral part of software development, as its popularity and adoption continues to grow according to reports [9] [8] [5]. However, reasonable uncertainty still surrounds its implementation in practice. It becomes imperative to investigate the implementation of the concept in practice. We seeks to answer the research question: "How do practitioners in the study perceive and implement DevOps in practice?" In particular, we will address the following questions:

- RQ1:What are practitioners' perceptions of DevOps definition and description?
- RQ2:How is DevOps implemented in practice?
- RQ3:How are DevOps functions different from IT Operations and development teams functions?

To answer the research questions, we conducted an empirical case study based on interviews of practitioners. The research adopts a method informed by Grounded Theory, which is a systematic methodology of qualitative data gathering and analysis, aimed at theory construction which allows for the emergence of new concepts grounded in the data [10]–[12]. This is mainly due to scarcity of literature on DevOps implementation in practice. [13] stated that "Grounded Theory is an excellent method for studying software engineering and generating theories that are relevant to practitioners". Their study provides a mode for using Glaserian Grounded Theory in software engineering research, which we have also adopted in our research. Other software development investigations

We are grateful to Petroleum Technology Development Fund (PTDF) for funding this research at the University of Salford, UK

have also demonstrated that Grounded Theory is well suited to study practitioners interaction with the concept of DevOps ([14] [15] [4]). We identified four approaches to DevOps implementation in the study, mapped them to on-premises and cloud-based deployments, and identified the DevOps practices facilitators for each of the approaches from practitioners' point of view. To the best of the authors' knowledge, no previous study explores DevOps in this regard. The rest of this paper is organised as follows: we explore the concept of DevOps in literature, presenting views of its description, characteristics, and scope. We continue by describing the study methodology, followed by a section on our findings. We discuss the findings and present a conclusion.

II. The Nature of DevOps

DevOps is described as a software engineering culture and philosophy that utilises cross-functional teams, to build test and release software faster and more reliably through automation. It aims at better collaboration between development and operation team in software development. According to available literature, benefit derived from DevOps adoption include faster delivery, improved quality and security, and better collaboration [3] [4] [16]. In [3], it was noticed that "teams are happier and more engaged", and shared technical knowledge between Ops and development teams increased collaboration between them after DevOps implementation. There are however differing explanations of what the DevOps concept means. Smeads et al [17] contend that there are two conflicting views of DevOps in the blog-sphere: the view that DevOps is a cultural movement to facilitate rapid software development and deployment, and the argument that it is rather a job description requiring both development and IT operation skills. While the former view is more predominant in available literature, the second view is mostly alluded to [4] [8] [17]–[19]. However, Humble [20] stated in his blog post that "there's no such thing as a 'DevOps team'". Huttermann [21] also insisted that DevOps is neither a job description nor a department or a unit in an organisation. He stated that "DevOps describes practices that streamline the software delivery process, emphasising the learning by streamlining feedback from production to development and improving the cycle time". Agreeing with Huttermann's stance [21], [22] identifies four significant characteristics of DevOps in a blog post as culture, automation, measurement and sharing which was referenced in [8]. Yet, the 2014 State of DevOps report shows a growing number of DevOps teams [9]. Although tailoring of software methods to suit specific organisational need is common practice [23], and it is argued that organisations should choose their own approach to DevOps, the ambiguity and conflicts in the description of DevOps has resulted in uncertainty of how to effectively implement the concept [24] [15] [4]. A road-map for the standardisation of DevOps terminologies and practices was presented in [25]. [15] carried out both a SLR and qualitative study of DevOps application in practice based on data from six organisation. The study mentioned four ways of DevOps definition in literature. It briefly described DevOps adoption of the individual organisations but did not provide details of actual implementation of the concepts in these organisations. [8] mentioned three possible approaches to DevOps, and provided evidence of investigation into one of them – mixing responsibilities between development and operations. The study identified improved collaboration and trust, shared responsibility, and smoother workflow as benefits of the approach. A systematic literature review (SLR) [24] used ISO/IEC 24714 metamodel elements to sort DevOps into people, process, technology, and artefacts. The study reported the occurrence of DevOps Engineer role, which can be performed by either developers or Ops specialists, and identified automation as an enabler of DevOps. In addition, the scope of DevOps differs in its adoption. While some organisations view DevOps as just "deployment automation" by select cross-functional teams, others like International Business Machines (IBM) consider it to be "improved automation, integration, collaboration, and optimisation of development and operations" [7]. Although there are no prescribed ceremonies in DevOps, it advocates practices like continuous integration, continuous delivery, continuous deployment, automated testing, infrastructure-as-code, and automated releases.

There are however challenges like the lack of appropriate skill-sets, fast-evolution of technology stack and tools, as well as resistance to change, associated with the DevOps adoption journey [3] [4] [16]. [24] identified people and culture, and the misunderstanding of DevOps as major challenges. [8] stated that mixing of responsibilities between developers and Ops specialists has created a new source of friction between both groups of stakeholders.

Beyond anecdotal evidence [26] and survey data however, empirical study of the impact of DevOps on the software value stream is scarce in literature [15]. [4] carried out an exploratory case study to provide empirical evidence of the impact of DevOps adoption in a New Zealand organisation, where DevOps was perceived as "embedded Ops", as dedicated Ops specialists were part of development teams. They identified the employment of automation and cross-functional teams as facilitators of DevOps value delivery. [27] introduced a flow framework to create a value stream integration diagram which he said would give perspective to how DevOps worked in practice.

Our area of interest lies mainly in the understanding of DevOps implementation among industry practitioners. As literature that provide empirical evidence on the variants of DevOps implementation in practice is scarce, the research adopts an exploratory case study approach to investigate DevOps implementation in industrial settings.

III. RESEARCH METHODS

The methodology and sequence of actions taken in the study is described in this section. First, a background to the case study is given, then data collection and thereafter, the data analysis is explained. We carried out an exploratory empirical study based mainly on interviews with industry

practitioners, and adopted an approach informed by Grounded Theory [10] [28]in our data analysis. For basic understanding of the concept, the authors engaged in light literature review as prescribed in Glaserian Grounded Theory [11] [12]. This facilitated effective discussions during interviews. However, in-depth study was done to understand and carry out Grounded Theory methodology.

A. Research Sites

Eleven practitioners from nine organisations participated in the study. The primary unit of analysis is software development practitioners with at least two years experience of using DevOps practices. Recruitment of participants for the study was matched with the primary unit of analysis. Organisations represented in the study are software development companies as well as software intensive companies, in the financial and public sectors based in the UK, Netherlands and Africa. The diversity in the research sites provides richness to the data and lends credence to the results. For instance, one of the participating organisations is a multinational bank in the Netherlands that deliver services to corporations and other financial institutions through in-house solutions deployed both on-premises and cloud-based platforms. The organisation is one of the largest in the world and has significant presence in Europe, with office around the world. Another is a large international software development company based in Africa. Table 1 shows a summary of their description. Some had co-located teams, while others were geographically distributed. Classification of organisations in this study is based on staff count, adopted from the EU Recommendation 2003/361.

B. Data Collection

The source of data collection was through interviews with 11 practitioners across nine organisations (as shown in Table 1), conducted over a period of four months in 2019. The study begins with the technique of initial purposive sampling, due to the difficulty in getting organisations to participate in research. Each participant was given an information sheet which told them that the interviews will be recorded, and consent form on which they indicated their choice of anonymity. Interviews were conducted over Skype and lasted an average of 45 minutes. A bespoke semi-structured interview guide [29] was designed, which contained a range of open-ended questions relating to practitioners' perception and practice of DevOps. Initial questions were generated from both experience with investigating agile methods, and light literature review of DevOps. This passed through several iterations of reviews by both researchers and were modified and evolved as data collection progressed following constant comparison process. The questions were tailored to suit interviewee's role [30] during interviews. A summary of participants' description is shown in Table 2. Recording of the interviews were transcribed, and transferred to Nvivo, a qualitative analysis software.

C. Data Analysis

All Interviews were transcribed verbatim to avoid distortions in meanings [30] [31]. This study involved four main aspects of data analysis as stipulated by Glaserian Grounded Theory: open coding, memoing, constant comparison, and saturation.

1) Open Coding: We began with the identification of concepts found within the interview transcripts [10], which involved line-by-line coding of participants' responses without any pre-determined codes. The authors used brief descriptive phrases to represent codes. In the first instance, the codes were handwritten onto hard copies of the interview transcripts, producing 21 codes. A second transcript was independently coded using the Nvivo software, from which 28 codes emerged. After an initial comparison of the two independent transcripts, the codes were merged into a single set of codes. These were preliminary codes which evolved as data analysis progresses. Subsequent transcripts are coded with already identified codes as well as newly identified codes. Data was then grouped into categories using concept classification [11], which become saturated as new data is added [10].

2) Memoing: In this research, we used memos to capture and refine concepts, and to express the relationship between concepts identified using open coding as they develop into categories [11]. Based on the identified codes, brief notes on topics were made containing quotations from transcripts as primary evidence, from which 10 memos emerged. Memo writing helped to elucidate and converge ideas originating from the codes, and sharpen categories evolving as new transcript data is added [10]. This is a key stage in theory generation [13]. Fig. 1 is an example of the emergence of memos from codes and categories generated from the data.

3) Constant Comparison: We used constant comparison technique to iterate between data collection and analysis, constantly comparing data within itself and other instances of the same event, without any preconceived outline. The technique was used to refine categories and their roperties, define and write the theory [10] [13].

4) Saturation: As expected, evidence began to converge at the later stage of the research, and the addition of new interviews had less and less impact on the categorisation. Saturation is said to have occurred in the research when new categories no longer emerge. There are arguments for theoretical and data saturation, and some hybrid forms. While the aim of the study is not to draw distinction between theoretical and data saturation, it adopts Glaser's definition of saturation: "Saturation means that no additional data are

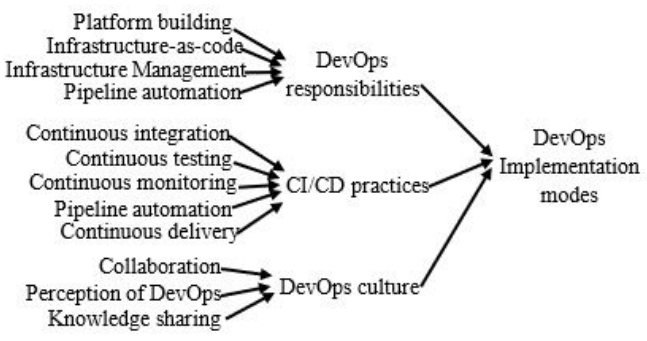


Fig. 1: Memo generation process

TABLE I: Description of organisations in the study

Organisation	FinCo1	FinCo2	FinCo3	FinCo4	ITCo	RegCo	FreeCo	PubCo	FinCo5
Size	Large	SME	SME	Large	Large	SME	SME	Large	SME
Business Type	Financial	Financial	Insurance	Financial	IT Consulting	Regulatory	IT Consulting	Public	Financial
Team Location	Distributed	Co-located	Co-located	Distributed	Distributed	Co-located	Co-located	Co-located	Co-located
Team Types	Developers DevOps Ops	Developers DevOps Ops	Developers DevOps Ops	Developers DevOps Ops	Developers DevOps	Developers DevOps	Developers Ops	Developers Ops	Developers
Tools	GitHub, Kubernetes, Ansible, Docker, Azure DevOps, Terraform, Istio	GitHub, SonarCube, Docker, Azure DevOps, Selenuim, Veracode, Slack	Kubernetes, GitLab, Ansible, Docker, Bamboo, Jenkin, Terraform, Vault	Jenkins, Jira, Slack, Gitlab	Github, AWS cloud formation and other AWS tools, New relic, Slack, Terraform	Gitlab, Slack, Kibana, Grafana, Jenkins, Jira, Terraform	Azure DevOps, Skype, Slack, Github	-	Github, Docker, Ansible, Azure DevOps, Terraform, Slack
Practices	Scrum meetings CI/CD	Scrum meetings CI/CD	Scrum meetings CI/CD	Scrum meetings CI/CD	Scrum meetings CI/CD	Scrum meetings CI/CD	Scrum meetings CI/CD	Scrum meetings	Scrum meetings CI/CD
Software Methodology	Scrum, Spotify	Scrum	Scrum, Kanban	Scrum, Spotify	Scrum	Scrum, Kanban	Scrum	Scrum	Scrum

TABLE II: Participants' description

Code	Job Title	SD Experience (years)	DevOps practice (years)
Finco1_DOps1	DevOps Engineer	18	4
Finco2_DOps	DevOps Engineer	12	7
Finco1_DOps2	DevOps Engineer	20	4
Finco3_DOps	DevOps Engineer	7	3
Finco6_DOps	Network/DevOps Engineer	31	5
ITco_DOps	DevOps Engineer	9	3
Finco4_Dvr	System Developer	14	3
Finco4_Dvr	Software Engineer	3	3
Regco_Dvr	Software Developer	15	2
Freeco_Prjm	Project Manager	20	4
Pubco_Dba1	Oracle Apex Developer	9	-
Pubco_Dba2	Oracle Apex Developer	19	-

being found" [10]. Emerging theories were identified from memos of transcripts, more data is collected and coded and constantly compared with the existing codes and memo until no additional information seemed apparent. Although this is not a claim of saturation, we believe that at this point in our work, we can examine the theory emerging from analysis of the data.

IV. Findings

Our findings from analysis of interview transcripts is described in this section. First, we present the description of DevOps. This is followed by approaches DevOps implementation, then we explore the DevOps Engineer/team responsibilities. Thereafter, we present the identified difference between developers, DevOps, and IT Ops teams. Out of the nine organisations that participated in the study, eight had adopted DevOps culture and implemented its practices across the organisation. The remaining organisation has thus been excluded from the finding herein presented.

A. Description of DevOps

Generally, participants described DevOps as better collaboration between developers and Ops teams. Some interviewees also described DevOps as end-to-end automation of the software development pipeline, providing better software quality, and creating seamless workflow of products at the shortest possible time. We divided these descriptions into two groups: DevOps as a culture, and as a job description.

1) DevOps as a Culture: The culture of collaboration between developers and Ops specialists was widely reported among interviewers. In FinCo1, the DevOps team assist developers re-configure their codes for containerisation. "And so we joined with the team and we told them how we're actually working. And together with them, we tried to, we need to adjust the application because it was not container aware. So, together with them, we altered the code a little bit, so it was container-ready". [Finco1_DOps1]. Knowledge is shared and a mutual understanding of basic activities across the boundaries of teams, achieved through collaborative resolution of code-related challenges. Similarly, [Finco1_DOps2] mentioned knowledge flow from developers to DevOps Engineers: "So as I mentioned, there's a couple of proper developers, we have a great resource because they teach us, nope, sit back, look. What are you trying to achieve? And write it properly from scratch rather than just couple together something".

DevOps teams in these organisations understand the codes of developers and help out where necessary. Developers are also made aware of how the automated infrastructure works, though not directly involved in its creation or maintenance. According to some practitioners, a level of confidence brought about by the basic understanding of other aspects of the process and familiarity with the other actors. Intra-team collaboration is reported as brainstorming and coding together when issues are encountered. Collaboration in FinCo2 involves the DevOps team creating users' stories from requirements, breaking them into manageable tasks and delegating these tasks to developers through Azure DevOps.

2) DevOps as a Job Description: Some of the interviewees had the job title of 'DevOps Engineer' and worked in distinct DevOps teams or departments. "We don't actually have developers in our team. So, in our case… it's just DevOps" [Finco1_DOps1]. They further described their team as "platform builders" for developers, "who support them and host their applications on our platform." Here, we see DevOps being presented as a job description, with DevOps Engineers responsible for carrying out "DevOps functions", which we

TABLE III: DevOps implementation approaches in the study

Mode	Ops	Outsourced-Ops	DevOps	DevOps bridge
Organisation	FreeCo1	FinCo5	ITCo1, RegCo1	FinCo1, FinCo2, FinCo3, FinCo4
Deployment platform	Hybrid cloud	Cloud	Cloud	Hybrid cloud

describe in the DevOps teams responsibility subsection. Some participants expressed concern that having separate DevOps teams might not be the right way to implement the concept, however, others affirmed that the approach allowed developers focus on providing value for the business, as they are not encumbered with the extra burden of creating and managing automated CI/CD pipeline for their deployments.

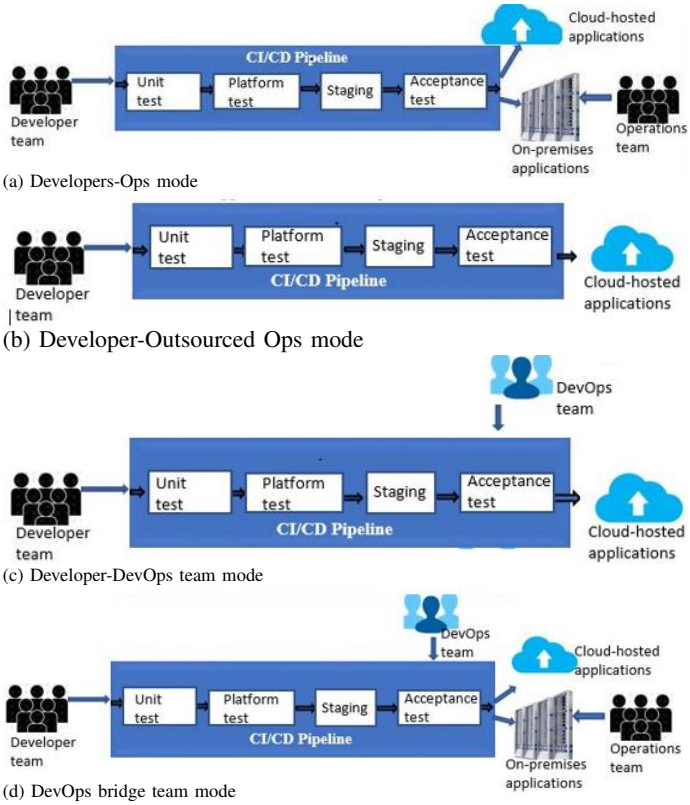
B. Approaches to DevOps

Practitioners in the study tell us that the decision of an effective approach to DevOps implementation remains largely uninformed as there are no prescribed best practices. From our analysis, we present a taxonomy of DevOps implementation: four dissimilar modes describing developers interaction with Ops, Outsourced Ops, DevOps, and DevOps Bridge teams. Fig. 2 shows the description of these modes. In Table 3, we show the distribution of the organisations according to their modes of DevOps implementation. It is important to note that the developers' teams encountered in the study include QA and security experts.

Developers-Ops mode (shown in Fig. 2a) of DevOps implementation depicts the instance where senior developers performed automated infrastructure management alongside development activities in a hybrid cloud deployment environment. The IT Ops team support the physical infrastructure and on-premises hosted application, while developers wrote application codes, deployed their codes through CI/CD pipelines, and managed applications themselves. Here, senior developers were seen as the facilitators of DevOps practices.

The Developers-Outsourced Ops mode (shown in Fig. 2b) is similar to Developers-Ops mode described above. Senior developers also write infrastructure codes to create and manage deployment pipelines, the difference being that its deployment environment is cloud-based, eliminating the need for Ops experts.

Fig. 2c depicts the Developers-DevOps mode where DevOps teams creates, deploys, and manages both the cloud infrastructure and deployment pipelines. Developers applications are also deployed and maintained by the DevOps team. Describing this mode, [Regco_Dvr] said "it allowed developers to focus on providing value for the business". This claim was clearly expressed by some other participants. Here, developers are not responsible for application deployment and management. Completed applications or features are handed over to the DevOps teams for deployment and management, who are the DevOps practices facilitators. DevOps bridge team mode was the mode widely used in our study. This mode (shown in Fig. 2d) was found in hybrid environment of cloud and on-premises deployment. Here, DevOps teams interface with both developers and IT Ops to drive the practices of DevOps like



(a) Developers-Ops mode

(b) Developer-Outsourced Ops mode

(c) Developer-DevOps team mode

(d) DevOps bridge team mode

Fig. 2: Pictorial representation of DevOps implementation

configuration management, continuous integration and continuous delivery, automated testing, deployment, monitoring, and metrics collection. Developers provide business solution, leaving the creation, deployment, and management of both the cloud infrastructure, virtual systems, and deployment pipelines to the DevOps teams. These teams are provided services of on-premises infrastructure by the Ops team. Essentially, the DevOps teams are customers to the Ops team, and service providers to developers. It is important to note that in this approach to DevOps, everyone is responsible for their actions. Developers create codes, deploy them through CI/CD pipelines, monitor and manage applications. In the same vein, the DevOps team deploy their infrastructure through the pipelines they create. However, the bridge teams are the facilitators of the DevOps process. Further investigation to understand this approach revealed three main classes of activities: provisioning and maintenance of physical systems, function virtualisation and creation of automated pipelines, and development, deployment and maintenance of applications (Continuous practices).

At the physical level, participants clearly explained that the systems in their on-premises data-centers needed to be configured and managed by IT Ops teams. Access is thereafter granted to DevOps engineer, who built automation into these systems. [Finco1_DOps2] describes the situation as "There's one team that does the physical installation, cabling of the hardware. There's another team that does the installation of some sort of the OS, and there's another team that customises that OS and we get access to install our [tools]... So, we are

the consumers of it."

Using automation tools, DevOps engineers create pipelines to enable continuous practices such as continuous integration/continuous deployment, continuous testing etc. Scripts are created for configuration management, and the deployment of infrastructure-as-code. In FinCo1, FinCo2, FinCo3, and FinCo4, these activities are solely the responsibility of the DevOps teams. Applications and solutions are subsequently developed and deployed through automated pipelines. The processes are mostly automated, however code review and user acceptance tests were described as manual processes.

Fig. 3 is a quadrant showing a taxonomy of DevOps implementation identified in the study. It describes the interaction of developers with various teams and presents a summary of the activities in each mode of DevOps implementation. The x-axis shows the application deployment environment. The y-axis shows the facilitators of DevOps practices.

Despite the seeming prevalence of bridge teams in the study, some interviewees thought it was not the right approach to DevOps implementation, as "there is still segregation between development and infrastructure in some way." [Finco1_DOps1]

Although tooling is considered an essential part of DevOps, we have not explored it in this paper as our focus is mainly on the approach to its implementation.

C. DevOps Teams' Responsibilities

The DevOps teams under study are be tasked with migration from existing platforms to either cloud based or an automated on-premises environment, and its subsequent maintenance. Generally, they act as intermediary between IT Ops and developers, providing the means to an end in software development (SD) by creating automated pipelines on both physical and virtualized servers to enable continuous integration and continuous delivery. In FinCo1 and ITCo1, DevOps teams are organised around specific products. Beyond provisioning automated platforms and maintenance of the environment however, there are only slight variations in the responsibilities

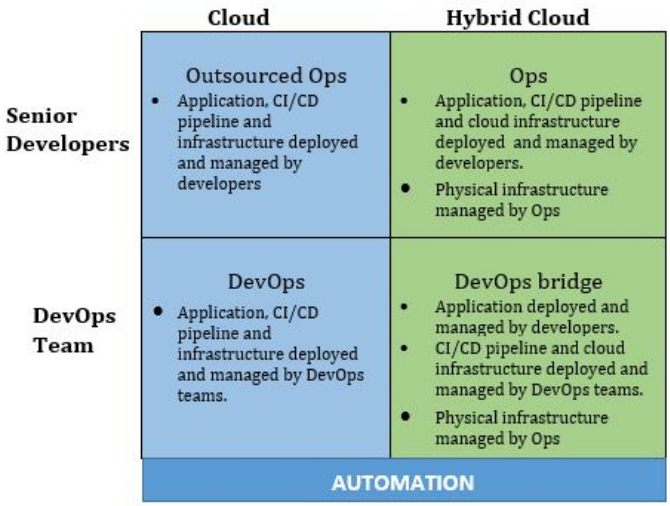


Fig. 3: Taxonomy of DevOps Implementation in the study

of the DevOps teams encountered in the study. For example, the DevOps team in FinCo2 is tasked with development cycle automation and tools unification. They also coordinate the activities of the software development process.

In RegCo1 and ITCo1, DevOps teams are responsible for all deployments. Developers hand over applications to these teams, who oversee the journey through the CI/CD pipeline. The teams also monitor the applications and function as first line of support.

D. Functions Variations between Developers, DevOps Engineers, and Ops specialists

An interesting point observed is the insistence of a distinction between "pure developers" and the DevOps engineers and IT Ops by participants. For instance, some DevOps engineers interviewed constantly referred to developers as "them". Some participants expressed the view that software developers should spend their time working on their products and not be bothered with what goes on "beneath the hood" of the infrastructure side of the platform, as long as they can deploy solutions seamlessly.

"Yes, they're purely development teams. They create APIs, beacon APIs or a full-term application or just parts of websites...The only thing that we actually want them to do is write the code and we should know how to start your application. And they create the docker file which comes with it... from there we try to pick up the rest." [Finco1_DOps1]

Also, while the DevOps team works with the goal of giving developers the best tools to get their work done, they expect developers to take responsibility for their products. This suggests boundaries of responsibilities.

Another observed instance is the distinction between DevOps and IT Ops, clearly seen in the responsibility for product codes and delivery, the deployment pipeline and automated infrastructure management, and the physical infrastructure administration. Table 4 summarizes activities of developers, DevOps and IT Ops teams identified in the study.

All deployments in the study, from developers and DevOps teams, go through automated pipelines. Here, we see an intersection of an activity (deployment) between the developers and DevOps team, however, Table 4 creates distinction between the types of deployments carried out by each group.

V. DISCUSSION

In this section, we discuss the findings in relation to the research questions and implication of research.

A. Description of DevOps (RQ1)

To answer the questions "What is practitioners' perception of DevOps definition and description?", our findings show DevOps being described by practitioners in the study as not just a culture and specific job description, but also distinct teams separate from both developers and IT Ops teams. Although members of these teams have backgrounds in either software development or IT Ops, the nomenclature "DevOps" now separates them from their original silos and

TABLE IV: Difference between developers, DevOps and IT Ops

	Developers	DevOps Teams	IT Ops
Coding	Write code for products and features	Write codes for tools and virtualized functions	Write scripts for functions
Continuous Integration	Use CI/CD pipeline to continuously merge code to master branch	Use CI/CD pipeline to continuously merge code to master branch	-
Deployment	Deploys own solutions to both cloud and on-premises platforms using automated tools	Deploys own solutions to both cloud and on-premises platforms using automated tools	Deploys developers' solutions to on-premises physical and virtualized systems
Infrastructure management	Manages cloud infrastructure (1 instance)	Automation pipeline management on both cloud and on-premises infrastructure	Physical infrastructure management
Other identified responsibilities (1 instance)	-	Translates requirements to user stories, scrum masters, assigning tasks to developers, project tracking/monitoring	-

classifies them as a unique team of "platform builders" as Finco1_DOps1 described it. This seems consistent with the 2014 State of DevOps report of a growing number of DevOps teams [9]. In [20], Humble mentioned that Ops teams may sometimes be referred to as "DevOps team" in some organisations when they build self-service platforms, provide tool chain, training and support developers and the platforms. However, he argues strongly that the insertion of "another layer of indirection between the dev and ops team and call it a 'Devops team'" is not the right way to address the existing gap. Intriguingly, most teams under study fit nicely into both sides of Humble's argument - they are teams between developers and IT Ops teams who provide all the above-mentioned services.

B. Implementation of DevOps Practice (RQ2)

To answer the question "How is DevOps implemented in practice?", we present a taxonomy of four approaches to DevOps based on interviewees description of the concept and practice. As described in the finding section of this paper, they are developers' interaction with On-premises Ops, Outsourced Ops, DevOps teams, and DevOps bridge teams. This describes the collaboration between developers and DevOps teams based on automation, mixed responsibilities of developers without an IT Ops team, developers working with DevOps teams only, and DevOps teams serving as a bridge between developers and IT Ops teams. These are similar to [8] suggestion of three possible approaches of mixed responsibilities, mixed personnel, and bridge team, in which they provided evidence and analysed one of the said approaches (mixed responsibilities). This study however differs from theirs in the following ways: firstly, we provide empirical data confirming practitioners' implementation of all three approaches mentioned in their studies. Secondly, we present a novel mapping of the approaches to on-premises and cloud-based deployments, thereby presenting a fourth approach - which is essentially a variant of mixed responsibilities but can only be found in cloud-based deployment environment. Thirdly, we identified the "facilitators" of DevOps practices in the four approaches identified in the study.

C. Functions Variations between Developers, DevOps Engineers, and Ops Specialists (RQ3)

To answer the question "How is DevOps functions different from IT Operations and development teams' functions?", we described the functions in DevOps practices identified in the study. Generally, the responsibilities from development to deployment include creating a piece of software, testing, building and maintenance of automated deployment pipelines, integration, deployment, providing metrics, and platform migration. However, personnel carrying out these functions vary, depending on the DevOps approach adopted by an organisation. The distinction in functions was particularly seen in the Developers-DevOps-Ops mode, which from Table 3, seem more prevalent in the study. Four out of the Five organisations with hybrid platforms have teams made solely of 'DevOps engineers'. In a nutshell, IT Ops teams provide services to DevOps teams in the form of physical infrastructure, DevOps teams provide services to developers in the form of tooling and automated pipelines, while developers provide business value in the form of applications, features etc.

D. Threat to Validity

The study adopts the assessment for rigour and trustworthiness in qualitative research as proposed by Lincoln and Guba [32]: confirmability, dependability, internal consistency (credibility), and transferability.

1) Confirmability: Confirmability denotes the researcher's ability to present a chain of evidence showing that the data is a true representation of participants' responses, and not the researcher's bias or opinion [13] [33] [34]. To ensure confirmability, we described how conclusions and interpretations in the study was established.

2) Dependability: Dependability refers to the repeatability of the study [13] [33] [34].The exact methods of data collection, data analysis and interpretation are clearly described in the paper. A sample of interview questions is included in this paper.

3) Internal Consistency: Internal consistency indicates credibility and consistency of finding [13] [33] [34]. Participants' responses are clearly described. Quotes from participants are also included in the report.

4) Transferability: Transferability describes the applicability of the study to other settings [33] [34], and useful theories can be derived from findings [13]. This criterion may be relevant in a qualitative study, depending on the inclination of the research. The study is exploratory. The concern being an in-depth investigation into the practice presented. We make no claim to creating a statistically significant survey.

VI. CONCLUSION

DevOps is described as a cultural movement in software engineering aimed at building, testing and releasing software

faster and more reliably through automation. According to available literature, benefits derived from DevOps adoption include faster delivery, improved quality and security, and better collaboration. However, there are differing descriptions of the concept. While some studies characterise it as an organisation-wide culture to foster better collaboration between IT Ops and development teams, DevOps has also been portrayed as a job description by others.

To understand its implementation in practice, our paper presents an exploratory study based on interviews with 11 DevOps practitioners across nine organisations, with data analysis informed by Grounded Theory. The empirical findings in this paper provide new understanding of DevOps implementation, based on the descriptions by the practitioners in our study. Firstly, this study provides in-depth analysis and presents a taxonomy of DevOps implementation in practice. We show four modes of its implementation, categorised as developers' interactions with: Ops, Outsourced Ops, DevOps teams, and DevOps bridge teams. Secondly, the paper presents a novel mapping of these approaches to cloud and on-premises deployment environments. Thirdly, we identified the facilitators of DevOps practices in these modes. Finally, further analysis of the fourth approach exposed three distinct groups of activities: provisioning and maintenance of physical systems, function virtualisation and creation of automated pipelines, and development, deployment and maintenance of applications. This, we believe, may have given rise to the implementation of DevOps as teams between developers and operations teams, as each group of activities require specific skill-sets. Based on the study, we conclude that DevOps is perceived by participants as both a culture and a job description, and these two views are not necessarily mutually exclusive. Also, the different modes of DevOps implementation seems driven by other organisational factors beyond its perception. While the practices appear the same across the modes of implementation, functions and responsibilities differ. This may be a source of variation in DevOps value realisation.

REFERENCES

[1] L. Bass, I. Weber, and L. Zhu, *DevOps: A software architect's perspective*. Addison-Wesley Professional, 2015.

[2] A. Dyck, R. Penners, and H. Lichter, "Towards definitions for release engineering and devops," in *2015 IEEE/ACM 3rd International Workshop on Release Engineering*. IEEE, 2015, pp. 3–3.

[3] G. Kim, J. Humble, P. Debois, and J. Willis, *The DevOps Handbook:: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution, 2016.

[4] M. Senapathi, J. Buchan, and H. Osman, "Devops capabilities, practices, and challenges: Insights from a case study," in *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018*. ACM, 2018, pp. 57–67.

[5] K. Kuusinen, V. Balakumar, S. C. Jepsen, S. H. Larsen, T. A. Lemqvist, A. Muric, A. Ø. Nielsen, and O. Vestergaard, "A large agile organization on its journey towards devops," in *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2018, pp. 60–63.

[6] L. Bass, R. Jeffery, H. Wada, I. Weber, and L. Zhu, "Eliciting operations requirements for applications," in *Proceedings of the 1st International Workshop on Release Engineering*. IEEE Press, 2013, pp. 5–8.

[7] M. Rowe and P. Marshall. The business case for collaborative devops. [Online]. Available: https://www.ibm.com/developerworks/community/blogs/

[8] K. Nybom, J. Smeds, and I. Porres, "On the impact of mixing responsibilities between devs and ops," in *International Conference on Agile Software Development*. Springer, 2016, pp. 131–143.

[9] P. Labs. (2014) 2014 state of devops report. [Online]. Available: http://puppetlabs.com/sites/default/files/2014-state-of-devops-report.pdf

[10] B. Glaser and A. Strauss, "Grounded theory: The discovery of grounded theory," *Sociology the journal of the British sociological association*, vol. 12, no. 1, 1967.

[11] B. Glaser, "Theoretical sensitivity," *Advances in the methodology of grounded theory*, 1978.

[12] R. Hoda, J. Noble, and S. Marshall, "Using grounded theory to study the human aspects of software engineering," in *Human Aspects of Software Engineering*, 2010, pp. 1–2.

[13] S. Adolph, W. Hall, and P. Kruchten, "Using grounded theory to study the experience of software development," *Empirical Software Engineering*, vol. 16, no. 4, pp. 487–513, 2011.

[14] S. Bick, K. Spohrer, R. Hoda, A. Scheerer, and A. Heinzl, "Coordination challenges in large-scale software development: a case study of planning misalignment in hybrid settings," *IEEE Transactions on Software Engineering*, vol. 44, no. 10, pp. 932–950, 2017.

[15] F. M. Erich, C. Amrit, and M. Daneva, "A qualitative study of devops usage in practice," *Journal of Software: Evolution and Process*, vol. 29, no. 6, p. e1885, 2017.

[16] L. E. Lwakatare, T. Kilamo, T. Karvonen, T. Sauvola, V. Heikkilä, J. Itkonen, P. Kuvaja, T. Mikkonen, M. Oivo, and C. Lassenius, "Devops in practice: A multiple case study of five companies," *Information and Software Technology*, 2019.

[17] J. Smeds, K. Nybom, and I. Porres, "Devops: a definition and perceived adoption impediments," in *International Conference on Agile Software Development*. Springer, 2015, pp. 166–177.

[18] T. Clear, "Thinking issues meeting employers expectations of devops roles: can dispositions be taught?" *ACM Inroads*, vol. 8, no. 2, 2017.

[19] J. Roche, "Adopting devops practices in quality assurance," *Commun. ACM*, vol. 56, no. 11, 2013.

[20] J. Humble. (2012) There's no such thing as a devops team. [Online]. Available: https://continuousdelivery.com/2012/10/theres-no-such-thing-as-a-devops-team/

[21] M. Hüttermann, *DevOps for developers*. Apress, 2012.

[22] J. Willis. (2010) What devops means to me. [Online]. Available: https://blog.chef.io/what-devops-means-to-me

[23] J. M. Bass, "Artefacts and agile method tailoring in large-scale offshore software development programmes," *Information and Software Technology*, vol. 75, 2016.

[24] A. Q. Gill, A. Loumish, I. Riyat, and S. Han, "Devops for information management systems," *VINE Journal of Information and Knowledge Management Systems*, 2018.

[25] A. Wahaballa, O. Wahballa, M. Abdellatief, H. Xiong, and Z. Qin, "Toward unified devops model," in *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, 2015, pp. 211–214.

[26] M. Rembetsy and P. McDonnell, "Continuously deploying culture: Scaling culture at etsy," in *Velocity Europe*. O'Reilly, 2012.

[27] M. Kersten, "Mining the ground truth of enterprise toolchains," *IEEE Software*, no. 3, pp. 12–17, 2018.

[28] K.-J. Stol, P. Ralph, and B. Fitzgerald, "Grounded theory in software engineering research: a critical review and guidelines," in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. IEEE, 2016, pp. 120–131.

[29] J. M. Bass and R. W. Macarthy. (2020) Interview guide - devops (version 1). [Online]. Available: https://doi.org/10.17866/rd.salford.12349724.v1.

[30] B. J. Oates, *Researching information systems and computing*. Sage, 2005.

[31] S. Kowal and D. C. O'connell, "5.9 the transcription of conversations," *A Companion to*, p. 248, 2004.

[32] E. G. Guba, Y. S. Lincoln *et al.*, "Competing paradigms in qualitative research," *Handbook of qualitative research*, vol. 2, no. 163-194, p. 105, 1994.

[33] D. G. Cope, "Methods and meanings: credibility and trustworthiness of qualitative research." in *Oncology nursing forum*, vol. 41, no. 1, 2014.

[34] L. Krefting, "Rigor in qualitative research: The assessment of trustworthiness," *American journal of occupational therapy*, vol. 45, no. 3, pp. 214–222, 1991.