ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

ELISA COSTA DIEL

COMMUNICATION IN DEVOPS

Porto Alegre
2017

PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
SCHOOL OF TECHNOLOGY
COMPUTER SCIENCE GRADUATE PROGRAM

COMMUNICATION IN DEVOPS

ELISA COSTA DIEL

Thesis submitted to the Pontifical Catholic University of Rio Grande do Sul in partial fullfillment of the requirements for the degree of Master in Computer Science.

Advisor: Prof. Sabrina Marczak, PhD
Co-Advisor: Prof. Daniela S. Cruzes, PhD

Porto Alegre
2017

Ficha Catalográfica

D561c Diel, Elisa

Communication in DevOps / Elisa Diel . – 2017.
75 f.
Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientadora: Profa. Dra. Sabrina Marczak.
Co-orientadora: Profa. Dra. Daniela Cruzes.

1. DevOps. 2. Comunicação. 3. Estratégia de comunicação. 4. Condições de comunicação. 5. Estudo empírico. I. Marczak, Sabrina. II. Cruzes, Daniela. III. Título.

Elisa Costa Diel

Communication in DevOps

This Dissertation/Thesis has been submitted in partial fulfillment of the requirements for the degree of Doctor/Master of Computer Science, of the Graduate Program in Computer Science, School of Computer Science of the Pontifícia Universidade Católica do Rio Grande do Sul.

Sanctioned on 16th of March, 2017.

COMMITTEE MEMBERS:

Prof. Dr. Rafael Prickadnicki (PPGCC/PUCRS)

Prof. Dra. Maria Paasivaara (Aalto University)

Prof. Dra. Maya Daneva (Twente University)

Prof. Dra. Sabrina Marczak (PPGCC/PUCRS - Advisor)

Prof. Dra. Daniela S. Cruzes (SINTEF – Co-advisor)

“We keep moving forward, opening new doors,
and doing new things, because we’re curious
and curiosity keeps leading us down new paths.”
(Walt Disney)

ACKNOWLEDGMENTS

At the end of this journey, I wish to thank all of those who, somehow, have made the implementation of this work possible.

Above all, I thank my family for their understanding, unconditional love and for everything they represent in my life.

To my advisor and co-advisor, Sabrina and Daniela, for the meticulous, caring, and ambitious orientation and also for their constant incentives.

To all Teachers who collaborated with the teaching during the Masters and for the feedbacks about the research.

To all my research colleagues, for the discussions, contributions and supports throughout the period of the Master.

To all the friends and co-workers, for the help in the development of the research, for the understanding in the moments of resignation, and company in the moments of leisure.

To the participants in the course of the research, by the time dedicated and contribution.

To PUCRS and the Faculty of the Computer Science School, for all the infrastructure offered, not only during this work, but during all the years spent in the institution.

COMMUNICAÇÃO EM DEVOPS

RESUMO

Apesar de o Ágil buscar colaboração com todos as partes envolvidas, a maioria dos projetos ágeis não extende essa colaboração para o pessoal de operações. Problemas de comunicação são um problema recorrente em equipes ágeis que também é eminente na relação entre desenvolvedores e operações. Esta pesquisa visa compreender como a comunicação acontece em DevOps a partir das percepções dos praticantes. Para alcançar nosso objetivo, realizou-se uma Revisão de Literatura sobre DevOps e Comunicação, e conduziu-se um Estudo de Campo com dados qualitativos sendo coletados através de entrevistas. Os resultados indicam que hoje existem pelo menos três configurações diferentes de DevOps sendo aplicadas na indústria: profissionais Devs e Ops alocados na mesma equipe; uma equipe de DevOps com um conjunto de habilidades compartilhadas; e uma equipe separada de Dev e Ops trabalhando juntos. Apesar dessas configurações, não foram encontradas nenhuma particularidade. Em resumo, os resultados indicam que os membros de equipes co-alocadas e multi-funcionais se comunicam melhor; é importante trabalhar em conjunto e compartilhar conhecimentos técnicos; o poder de decisão variará de acordo com a situação enfrentada; entre outros. Nossas descobertas ajudam a diminuir a lacuna apontada por Erich, Amrit e Daneva entre Dev e Ops, avançando para uma melhor compreensão de como profissionais DevOps colaboram, ajudando eles a melhorar suas práticas de comunicação em seu trabalho diário.

Palavras-Chave: DevOps, comunicação, estratégia de comunicação, condições de comunicação, estudo empírico.

COMMUNICATION IN DEVOPS

ABSTRACT

Even though agile actively seeks collaboration with all kinds of stakeholders, most agile projects do not extend toward the operations people. Issues in communication are a recurring issue in agile teams. Such issues are also eminent in the relationship between developers and operations. This study aims to understand how communication happens in DevOps from the perspective of practitioners. To achieve our goal, a Literature Review on DevOps and Communication was performed, and a Field Study was conducted with qualitative data being collected through interviews. Results revealed that today there are at least three different DevOps configurations being applied in the industry, being: Dev and Ops professionals allocated to the same team; a team of DevOps with a shared skill set; and a separate team of Dev and Ops working together. Despite the configurations, we did not find any particularities. In summary, results show that co-allocated and cross-functional team members communicate better; it is important to work together as a single team and share technical knowledge; decision power changes based on the situation that is being faced; among others. Our findings help to narrow the gap pointed out by Erich, Amrit, and Daneva between Dev and Ops, moving towards a better understanding of how DevOps team collaborates by helping practitioners to improve their communication practices in their daily work.

Keywords: DevOps, communication, communication strategy, communication conditions, empirical study.

LIST OF FIGURES

Figure 2.1 – DevOps principles by França, Junior, and Travassos [20]. 19
Figure 2.2 – Communication Model by Mohr and Nevin [68]. 25
Figure 3.1 – Research design 28
Figure 3.2 – Phase 2 - Field study data collection and analysis process 30
Figure 3.3 – Thematic analysis steps by Cruzes and Dybå [17] 30
Figure 4.1 – Results: channel conditions - structure 34
Figure 4.2 – Results: channel conditions - climate 36
Figure 4.3 – Results: channel conditions - power 38
Figure 4.4 – Results: communication strategies - frequency 39
Figure 4.5 – Results: communication strategies - direction 41
Figure 4.6 – Results: communication strategies - modality 43
Figure 4.7 – Results: communication strategies - content 45
Figure 4.8 – Results: other findings - major factors for communication success . . 49
Figure 4.9 – Results: other findings - major challenges to successful communication 52

LIST OF TABLES

Table 2.1 – DevOps in a nutshell by Fitzpatrick et al. [37] 20
Table 2.2 – Communication types and techniques by Goles and Chin [39] 24
Table 4.1 – Background profile of the study participants 33
Table 6.1 – Summary of the findings 61

LIST OF ACRONYMS

DEV – Development / Developer

DEVOPS – Development and Operations

IS – Information Systems

IT – Information Technology

OPS – Operations

SE – Software Engineering

CONTENTS

1 INTRODUCTION . . . 13
1.1 MOTIVATION . . . 14
1.2 RESEARCH GOAL AND QUESTION . . . 14
1.3 RESEARCH METHODOLOGY . . . 15
1.4 MAIN RESULTS AND CONTRIBUTIONS . . . 15
1.5 DOCUMENT OUTLINE . . . 16

2 BACKGROUND . . . 18
2.1 DEVOPS . . . 18
2.1.1 DEVOPS AND COLLABORATION . . . 19
2.2 COMMUNICATION IN SOFTWARE ENGINEERING . . . 23
2.2.1 COMMUNICATION MODEL . . . 25

3 RESEARCH METHODOLOGY . . . 27
3.1 RESEARCH DESIGN . . . 27
3.1.1 PHASE 1: LITERATURE REVIEW . . . 27
3.1.2 PHASE 2: FIELD STUDY . . . 28

4 RESULTS . . . 32
4.1 PARTICIPANTS PROFILE . . . 32
4.2 CHANNEL CONDITIONS . . . 32
4.2.1 STRUCTURE . . . 33
4.2.2 CLIMATE . . . 35
4.2.3 POWER . . . 37
4.3 COMMUNICATION STRATEGIES . . . 39
4.3.1 FREQUENCY . . . 39
4.3.2 DIRECTION . . . 41
4.3.3 MODALITY . . . 43
4.3.4 CONTENT . . . 44
4.4 CHANNEL OUTCOMES . . . 46
4.4.1 COORDINATION . . . 46
4.4.2 SATISFACTION . . . 47
4.4.3 COMMITMENT . . . 47

4.4.4 PERFORMANCE 48
4.5 OTHER FINDINGS 49
4.5.1 MAJOR FACTORS FOR COMMUNICATION SUCCESS 49
4.5.2 MAJOR CHALLENGES TO SUCCESSFUL COMMUNICATION 51

5 DISCUSSION 54
5.1 DISCUSSION OF FINDINGS VS. EXISTING LITERATURE 54
5.1.1 WHAT IS A DEVOPS TEAM? 54
5.1.2 MAPPING THE COMMUNICATION PROCESS 56
5.1.3 RECOMMENDATIONS FOR INDUSTRY 58
5.2 CONTRIBUTIONS FOR RESEARCH 59
5.3 CONTRIBUTIONS FOR PRACTICE 60

6 FINAL CONSIDERATIONS 61
6.1 SUMMARY OF FINDINGS 61
6.2 PUBLICATIONS 63
6.3 LIMITATIONS 63
6.4 FUTURE WORK 63

REFERENCES 65

APPENDIX A – Script Interview **74**

APPENDIX B – Mind Map of the Results **75**

1. INTRODUCTION

Topics that are trends in industry and research today envision to improve the production line and work life-cycle of IT professionals adjusting to the growth in which the market has demanded. On that end, customers and users expect fast responses to their constantly changing requirements, concerning both functional and non-functional properties of a software [51]. To meet those expectations, companies have adopted some different strategies, with Agile Development being one of the methods most commonly used these days.

Even though agile actively seeks collaboration with all kinds of stakeholders, most agile projects still do not extend themselves towards the operations people. As organizations scale, so do development and operations teams, and while they may initially be co-located and have close communication links, increased team size and more strict separation of responsibilities can weaken such links [88]. The lack of collaboration between them as well as mismatch in configuration between development, testing, and production environment has made deploying software releases slow and painful for many organizations [43].

On an attempt to fix that, the industry has adopted the widely popular DevOps. The DevOps concept emerged from this increasing disconnect between the development and operations functions that arise within large software companies [36]. DevOps has its roots in the Agile software development movement and contributes to improvements and adoption of Agile practices, such as working software being the measure for progress, collaboration being a focus, and change being embraced [70]. Nonetheless, while this growing DevOps movement should be driving more effective partnerships and better integration opportunities, those are not coming fast enough to appease the significant IT market demand [58].

Note that, the promise of DevOps is not that it will reduce some costs or eliminate some internal hurdles, but that one can leverage valuable skills and knowledge to provide a product or service to one's customers [70]. Hence, DevOps represents the need to align the development and the deployment of that software into production [21]. Through the literature, it is possible to spot some gaps between developers and operations that DevOps tries to address. Shafer [81] and Thompson [27] call this a "Wall of Confusion" between IT development and operations. This "Wall" is caused by a combination of conflicting motivations among people, processes, and technology/tooling [28].

To overcome such a split that is predominant in many organizations today, organizational changes, cultural changes, and technical frameworks are required [94]. DevOps can come in handy in these scenarios by promoting this so needed cultural shift while modifying the interaction between developers and operations. However, for those cultural shifts to be effective, proper planning and a lot of effort are needed from all sides, since not only organizational but behavioral changes are also expected.

Brown and Starkey [7] argue that culture conditions attitudes towards communication, communication processes, and systems. Moreover, they complement that communication impacts directly on the decisions made by an organization. Therefore, reducing or eliminating this split between development and operation is of incredible importance to businesses as they seek to not only deliver new applications and features to the market as quickly as possible but also operationalize them at the stability and scale that the market demands [78].

1.1 Motivation

Software is created and released faster than ever, so the need for efficiency and integration between development and operations has become ever more important, giving the DevOps movement traction and visibility [78]. DevOps stresses improved communication, collaboration and interdependence between software development and IT operations [55] to facilitate the reduced cycle time, and, with multiple independent development teams, resulting in a need for continuous system integration and delivery methods [51]. To bridge these gaps, both teams will need to meet each other half-way regarding skills and collaboration and then work together once this sharing of knowledge and responsibility has been applied [78].

Sill [84] argues that earlier standards development practices were based on slower and more formal communication patterns that do not lend themselves to today's rapid progress and rapid cycling between conceiving new ideas and testing them in the field. He supplements his argument by stating that to alleviate this shortcoming, one needs to take a DevOps approach to bridge the gaps more quickly between formal ideas and practical implementation. In doing so, one also needs to scale the communication patterns horizontally to involve more opinions and feedback for the betterment of the field.

1.2 Research Goal and Question

Communication is an integral part of any relationship [87], and communication gaps is a recurrent problem in agile teams that is also eminent in the relationship between developers and operations. Therefore, the goal of this research is to characterize how the two-way communication happens in DevOps from the practitioners' perspectives, based on the communication model introduced by Mohr and Nevin [68]. To achieve that, the following research question was defined:

How does the two-way communication happen in a DevOps team?

To answer this research question, it was first necessary to understand the definition and the underlying concepts of DevOps, so that it would be possible to have a better understanding on the matter, being able to seek more depth information on the practices and processes of communication used by DevOps practitioners.

1.3 Research Methodology

To accomplish the posed research goal, a qualitative-based research approach was followed. The research process is organized in two phases, namely Literature Review and Field Study A Snowballing Literature Review [97] (Phase 1) about DevOps and Communication was conducted early in the research process in order to inform the field study design (Phase 2) and later updated for new studies.

For the Field Study [85], qualitative data were collected through interviews with DevOps professionals. After each interview, the voice recorded audio was transcribed and underwent a process of deductive [66] and inductive [14] coding. Once the coding was ready, a thematic analysis [17] was performed. Once the data were already analyzed, the validation of the same was performed.

Incremental feedback was collected upon the data analyzed. Feedbacks came in two forms of member checking [9]: (i) coding and analysis review with co-advisor; and (ii) seeking other active DevOps professionals opinions through the participation at 4 large conferences in the course of 2016: DevOpsDay Oslo [23] (guest speaker), ICGSE [56] (presentation of a short paper [24]), ComputerWorld Oslo [13] (guest speaker), and AgileBrazil [6] (topic approached in an open space).

1.4 Main Results and Contributions

The results indicate that today there are at least three different configurations of DevOps in industry. Those are: Dev and Ops professionals allocated to the same team; a team of DevOps with a shared skill set; and a separate team of Dev and Ops working together. Regardless of their team's structure, company size, or nationality, participants reported very similar outcomes. In summary, results show that:

- Work as one team: co-allocated and cross-functional team communicates better. To achieve better results in communication, it is important to work together (as one team), sharing technical expertises and leadership.

- Create a trusting environment: the decision-power will vary according to the situation. Dev and Ops should feel comfortable with that and jump in when needed. Sharing leadership and having empathy towards each other are key components for open and effective communication.
- Communicate everything: find the best communication tools that work for the team and the business needs. Then, allow people to create awareness and encourage them to contextualize all information. Share knowledge with as much frequency as necessary.

Besides, with the development of this work, the following contributions were achieved (Chapter 5 - Discussion - elaborates each in more detail):

C1. To approach and enrich a topic that is still considered new in academia.

C2. To use a communication theory in marketing, in an SE setting, to help construct the conceptual mapping on the subject.

C3. To present partial findings in large industry events as the form of content validation and saturation, strengthening the relationship between industry and academic research.

C4. To perform a cross-culture research with an expressive number of participants.

C5. To provide enough insights so that the industry can observe the different aspects of the subject.

C6. To identify challenges and solutions to assist the industry in improving their processes.

1.5 Document Outline

The organization of this document has been defined to facilitate its comprehension by the readers. After this introduction chapter, which describes the motivation, goals, and scope of the research, the document is organized as follows:

- **Chapter 2 - Background:** presents the theoretical background and literature review on the topic under the perspective of Software Engineering industry.
- **Chapter 3 - Research Methodology:** presents the research methodology, describing the adopted research, data collection, and analysis methods.
- **Chapter 4 - Results:** describes the key findings encountered by answering the posed research question.
- **Chapter 5 - Discussion:** discusses and synthesizes the findings, presenting the contributions for research and practice, and recommendations for software organizations.

- **Chapter 6 - Final Considerations:** presents an overall discussion of the conclusions, explores the major contributions, inherent limitations, and outlines possible future research in the area.

2. BACKGROUND

This chapter elaborates on the background and related work providing a comprehensive view of the foundations for this work. First, the chapter presents the concepts and different definitions of DevOps. Then, it discusses Communication in Software Engineering, concluding with the presentation of the Communication Model on which this research was based.

2.1 DevOps

There is a common sense among authors that DevOps is an emerging term that has yet to be further studied and defined. On the philosophical side, Hussaini [54] best defines DevOps as an acronym for Development (Dev) and Operations (Ops) of information technology systems and applications. He adds that the DevOps paradigm emerged as a response to the growing knowledge that there exists a gap of 4Cs (communication, cooperation, culture, and collaboration) between what is usually considered IT development function and IT operations function in an organization.

Debois [21] says that the term DevOps is only a stub for more global company collaboration, which he explains, works as follows: once priorities have been determined and work can start, developers pair together with operations people to get the job done. This pairing allows for better knowledge diffusion across the two traditionally separate teams. Issues such as stability, monitoring, and backup can be addressed immediately instead of being afterthoughts, and operations get a better understanding of how the application works before it is deployed into production. Also, feedback is available to all people: those in operations learn what issues they might expect in production while developers learn about the production environments. Feedback is not only of a technical nature – management and business can learn from production trial runs what customers want and how they react.

Hosono [49] brings a new focus into the definition of this concept by saying that the DevOps – an abbreviation term for development and operation – centers on two primary concepts: culture and technology. The culture seeks to change the dynamics, in which the development and operation teams interact with one another, emphasizing the tasks between design and operation, such as design for operation, test-driven development, and continuous integration. This culture is congruous to the technologies of tool chains, which are a collection of complimentary tools used to automate an end-to-end process. The tool chains enable lifecycle-based automation and rapid responses to changing business conditions and become more dynamically changeable via programmatic interfaces [42].

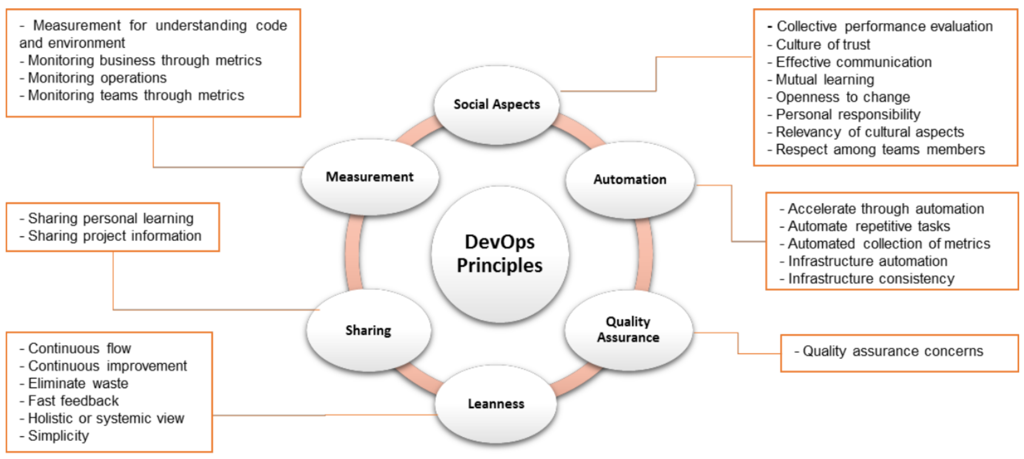

DevOps Principles
Social Aspects
Automation
Quality Assurance
Leanness
Sharing
Measurement
- Measurement for understanding code and environment
- Monitoring business through metrics
- Monitoring operations
- Monitoring teams through metrics
- Collective performance evaluation
- Culture of trust
- Effective communication
- Mutual learning
- Openness to change
- Personal responsibility
- Relevancy of cultural aspects
- Respect among teams members
- Accelerate through automation
- Automate repetitive tasks
- Automated collection of metrics
- Infrastructure automation
- Infrastructure consistency
- Quality assurance concerns
- Sharing personal learning
- Sharing project information
- Continuous flow
- Continuous improvement
- Eliminate waste
- Fast feedback
- Holistic or systemic view
- Simplicity

Figure 2.1 – DevOps principles by França, Junior, and Travassos [20].

Bang et al. [4] explains that DevOps has 4 perspectives: (i) a culture of collaboration between all team members; (ii) automation of build, deployment, and testing; (iii) measurement of process, value, cost, and technical metrics; and (iv) sharing of knowledge and tools. Complementing these perspectives, França, Junior, and Travassos [20] compiled a list of principles that characterize DevOps (see Figure 2.1), that being: (i) social aspects: interaction between Devs and Ops; (ii) automation: reduce unnecessary effort to improve software delivery; (iii) quality assurance: assuring the quality of both development and operations processes as products; (iv) leanness: the use of Lean Thinking [98] to help on the processes; (v) sharing: information and knowledge are disseminated among individuals to promote the exchange of personal learning and project information; and (vi) measurement: collect efficient metrics to support the decision-making.

Even though there are different understandings of what DevOps exactly means [30, 31, 86, 57, 20], in essence, they all want to achieve the same: to facilitate collaboration among developers and operations. Fitzpatrick et al. [37] presented an excellent overview of DevOps in a nutshell which summarizes most definitions available so far (see Table 2.1). For this research, the definition presented by Gottesheim [40] was chosen to be presented to our research participants due to its completeness and easy understanding, and also because it addresses not only Dev and Ops perspectives but it combines the notion of business into the concept as well: DevOps aligns business requirements with IT performance, with the goal of adopting practices that allow a fast flow of changes to a production environment, while maintaining a high level of stability, reliability, and performance in these systems.

2.1.1 DevOps and Collaboration

An essential requirement for DevOps success is to think of, prepare, implement, and support the new application as a mandatory component of a service that is being offered

Table 2.1 – DevOps in a nutshell by Fitzpatrick et al. [37]

Increased collaboration between operations and development
Reduced cycle times for operations activities (e.g., provisioning, deployment, change controls)
Extreme focus on automation in tools and processes
Fostering continuous improvement as both a means to the above and as a strategy to adapt to increasingly rapid changes happening in IT

to the business customer [58]. An essential part that which is implied within each of these success requirements is the communication process. Without it, to achieve greatness would be much harder. This section describes studies related to this topic.

Phifer [76] in 2011 wrote about the missing integration between development and operations and their conflicting priorities. While the former team is responsible for more releases, the latter is only worried about reducing and eliminating outages. He said that historically, many major outages and implementation failures result from a lack of communication between applications and infrastructure groups that do not participate effectively within a complete service management framework. Therefore, the DevOps movement is timely and relevant to try to solve this problem. Phifer believes that the key elements of DevOps to drive collaboration between development and operations can be achieved through additional process elements that can be found in the CMMI-DEV and ITIL together. He also mentions that to achieve success, it is also required to have an effective two-way communication between these groups (Devs and Ops) and the business representatives.

Also in 2011, Hosono et al. [48] proposed a cloud application framework which integrates both the development and production environment. To provide solutions for application development of the cloud in the view of the DevOps philosophy, they identified a set of causes of gaps between these two teams, such as "architecture design through combining functions" and "requirements and techniques for scalability." The framework identifies roles in development and phases in the development process for cloud applications. It also provides a set of design and evaluation tools for each phase, enabling effective development by sharing data among the phases. These patterned modeling tools and systematized practices reinforce the principle of developing cloud-based applications. They say that the framework, which is underpinned by the concept of life-cycle management for application, fills in the gap between the development and production phases of cloud application development.

In 2012, Shang [82] worked on bridging the gap between software developers and operations through the usage of logs. Back there, he said that developers do not usually divulge development knowledge about the software to operators, while operators rarely communicate field knowledge back to developers. He finds that bridging the gap between these two teams is essential because it would help to improve the quality and reduce the operational cost of large-scale software systems. He conducted a pilot empirical study on the execution logs of 10 releases of an open source software system named *Hadoop* and nine releases of a legacy enterprise application [83]. He proposed several techniques that make use of logs to support developers and operators such as: create field execution models from the field logs, attach the development history as well as bug reports to every log line, analyze the source code changed in a new release and we identify the log templates that are possibly impacted, among others.

Bruneo et al. [8] in 2013 introduced a framework - CloudWave. It uses the principles of DevOps to create an execution analytics cloud infrastructure where, through the use of programmable monitoring and online data abstraction, much more relevant information for the optimization of the ecosystem is obtained. Their motivation came from the perception that the transition to Cloud-based services demands both faster development cycles and the high degree of exploitation of the Cloud infrastructure. CloudWave comes in handy because its infrastructure is capable of monitoring the runtime environment, and through the use of execution analytics, both the infrastructure and applications can collaborate to provide dynamic reconfiguration. In addition to the automatic reconfiguration that the system can provide, developers are given access to a powerful SDK which implements the concept of DevOps to aid in shortening iterative development cycles. Therefore, CloudWave tries to promote communication, collaboration, and integration between developers and operational teams by shortening the loop between software creation, deployment, operation, and feedback.

Wettinger, Breitenbücher, and Leymann [95] believed, as Humble and Molesky [52] stated, that DevOps is an emerging paradigm to bridge this gap between these two groups, thereby enabling efficient collaboration. Furthermore, they believed that this gap could be bridge via the usage of practices and the correct tools. Therefore, in 2014 they presented a systematic classification of DevOps artifacts and showed how different kinds of artifacts could be transformed toward TOSCA (Topology and Orchestration Specification for Cloud Applications) through an automated transformation framework created by them. Their main motivation to this work was based on the fact that a huge number and variety of reusable DevOps artifacts are shared as open-source software. It was hard to combine and orchestrate artifacts of different kinds to automate the deployment of Cloud applications consisting of different middleware and application components. This automated framework enables the seamless and interoperable orchestration of arbitrary artifacts to model and deploy application topologies. They applied and validated the presented framework to implement transformation methods that generate reusable and interoperable TOSCA models for two different DevOps approaches Juju charms and Chef cookbooks (different cloud infrastructure automation tools).

Krusche and Alperowitz [63] endorsed that continuous delivery is a set of practices and principles to release software faster and more frequently. While it helps to bridge the gap between developers and operations for software in production, it can also improve the communication between developers and customers in the development phase, i.e. before software is in production. It shortens the feedback cycle and developers ideally use it right from the beginning of a software development project. In 2014, they reported on an implementation of a customized continuous delivery workflow and its benefits in a multi-customer project course in the Summer of 2013. Their workflow focused on the ability to deliver software with only a few clicks to the customer to obtain feedback as early as possible through the usage of an Issue Tracker and a central Continuous Integration Server.

Pérez, Wang, and Casale [75] worked in 2015 on a tool to fill in the gap between development and operations. Their point of view on DevOps is that DevOps is a novel trend in software engineering that aims at bridging the gap between development and operations, putting, in particular, the developer in greater control of deployment and application runtime. They considered the problem of designing a tool capable of providing feedback to the developer on the performance, reliability, and in general quality characteristics of the application at runtime. They state that the problem is therefore how software performance methods can help bridging the gap between run- time performance data and the higher level of abstraction required by the developer to be able to reason on the quality of an application design and possibly identify refactoring actions. So, the focus of their work was on designing the tool and identifying architecture and user requirements.

In 2015, Wahaballa et al. [93] proposed the Unified DevOps Model (UDOM) which can help mitigate problems caused by a conceptual deficit in the DevOps environments. Their definition of DevOps is that it is a software development method that advocates communication, collaboration and integration between software developers and operations teams to solve critical issues, such as fear of changes, risky deployments, blame game, and isolation. They add that this collaboration may cause a dangerous problem called a conceptual deficit. The Conceptual deficit comes from incomplete, wrong, or unimplemented nonfunctional requirements. The UDOM model includes three sub-models: application and data model, workflow execution model and infrastructure model, on which it unifies DevOps processes, terminology, concepts, patterns, cultures, and tools. With the main goal of the model being to assist the organizations to adapt to DevOps smoothly and easily.

Most of the studies on DevOps published in 2016 address the definition, adoption, and main challenges when adopting this approach [20, 57, 72, 77, 89]. Along with this, emerging technologies such as IoT (Internet Of Things), Cloud Computing, Large Scale Agile has also gained a lot of attention in this new scenario, since they all seem to have some impact on each other. Though, one thing they all have in common is that there is a pressing need for the harmonization of development and operations of an IT organization [54].

2.2 Communication in Software Engineering

In 1970, Frank Dance [19], a communication theorist, counted over 100 definitions of communication proposed by experts in the field. Now, even more definitions have emerged [99]. Duck and McMahan [25] say that many students assume that communication simply involves the sending of messages from one person to another through e-mails, phone calls, gestures, instant messages, text messages, or spoken word. Stating that while that basic view has some truth to it, communication involves a lot more than merely transmitting information from Person A to Person B.

Daiton and Zelley [18] explain that a common view of communication in the business world, for example, is that communication is synonymous with information. Companies want to hire and promote people with excellent communication skills. However, good communication means different things to different situations. Consequently, merely adopting a set of particular skills is not going to guarantee success. Good communicators are those who understand the underlying principles behind communication and can enact, appropriately and effectively, particular communication skills as the situation warrants. In complex situations, communication effectiveness is particularly critical to project success where multiple, and integrated stakeholder teams are involved, and where 'time to market' and project efficiency are key drivers [29].

In sharp contrast to the popular image of software developers as relatively introverted and isolated, they, in fact, spend a large proportion of their time communicating [46]. Espinosa and Carmel [32] argue that software development is a complex task with substantial non-routine, interdependent activities, which require a fair amount of communication to coordinate related tasks and work to be done. Coordination relies on communication, direct communication as well as communication mediated by code, documentation and artifacts [79]. It means that different people working on a common project agree to a common definition of what they are building, to share information, and to coordinate their activities [62].

Communication can occur in two ways: synchronously, where the exchange of information happens immediately, e.g., via chats; or asynchronously, where the exchange of information (the back and forth) takes longer to happen, e.g., e-mail exchanges. To that end, communication is also categorized into two different types (informal and formal), each playing an important part in the teams' organization. Goles and Chin [39] summarized the different communication types and main techniques of each type as presented in Table 2.2.

Informal communication refers to explicit communication via conversations among the workers in the companies [46] and is often based on existing communication technologies such as telephone, video, audio conference, voice mail, and e-mail [44, 87]. Consequently, face-to-face communication is presented as the best way to promote informal conversations. On the other side, Herbsleb and Mockus [46] and Kraut and Streeter [62]

Table 2.2 – Communication types and techniques by Goles and Chin [39]

Communication Types	Communication Techniques
Informal	Face-to-Face discussions in co-located or distributed teams [32] Informal discussions through telephone, video, audio conference, voice mail and e-mail [44, 87]
Formal	Group meetings such as weekly meetings, steering group/milestone meetings Status Meetings at which the personnel present the project results [74] Formal meetings through telephone, video, audio conference, voice mail and e-mail, progress reports [44, 87] Formal documentation, for example specification documents [46]

say that formal communication refers to explicit communication such as the specification documents, formal meetings, and the status review meetings in software development.

Even though, face-to-face communication is found to be the most effective among various channels of communication as it provides instant feedback and multiple cues like the expression, emotions, and personal focus [73, 16]. Mishra, Mishra, and Ostrovska [67] explain that the knowledge acquired through face-to-face communication can be contained for only a limited time, after which it starts to diminish gradually. Therefore, tools such as papers, white-boards, etc., may be used to store information planned for future use. They complement that these tools are also useful to access information about the project when many individuals are working together on one project, or when multiple teams (consisting of many individuals) are simultaneously working on different parts of the same project and coordination among them, hence, becomes paramount.

Smite [87] argues that communication is an integral part of any relationship, and as in any relationship, it can be problematic. To illustrate, Herbsleb and Grinter [45] present a telling example of poor communication in a global software development project, when a tester interpreted a "b-l-a-n-k" instruction as a space character instead of leaving the field empty, clearly not the intended message of the sender [11]. Etgar [33] suggests that conflict is caused by ineffective communication, which leads to misunderstandings, incorrect strategies, and mutual feelings of frustration [68]. Likewise, Khan et al. [60] highlighted some factors that can challenge communication, as follows:

- **Geographical Distance**: it is actually the effort required for one person to visit another. Generally, low geographical distance offers the high opportunity for people to communicate [47].
- **Socio-Cultural Distance**: it is a measure of a person's understanding of another person's values and normative practices [96]. Cultural distance involves national culture, organizational background, policies, and moral principles [47].
- **Temporal Distance**: it is the measure of time difference experienced by two people wishing to communicate [47]. Temporal distance results from different factors, including two people located in two different time zones, for example.

2.2.1 Communication Model

To help solve those problems introduced by Khan et al. [60], Tarone [91] brings the concept of Communication Strategies as a mutual attempt of two interlocutors to agree on a meaning in situations where linguistic and sociolinguistic structures do not seem to be shared. Therefore, it can be viewed as an attempt to bridge the gap between interlocutors in real communication situations. Mohr and Nevin [68] created a model for channel communication in marketing (see Figure 2.2) which explores the communication strategies of frequency, direction, modality, and content; channel conditions of structure, climate, and power; and channel outcomes of coordination, satisfaction, commitment, and performance. On their study, they developed a contingency theory in which the level of channel outcomes obtained is contingent upon an interaction between communication strategy and given channel conditions.

Extant Channel Conditions

- **Structure**: refers to the nature of the exchange relationship between parties - relational or discrete. Relational exchanges involve joint planning between parties; the relationship has a long-term orientation and interdependence is high. Discrete exchanges, in contrast, occur on an *ad hoc* basis - the relationship between parties has a short-term orientation and interdependence is low [65].
- **Climate**: refers to the feelings of the interlocutors about the level of trust and mutual supportiveness in the inter-organizational relationship [68]. Climate develops characteristics directly reflecting norms, leadership, and membership composition and provides a context for interpersonal communication [34].

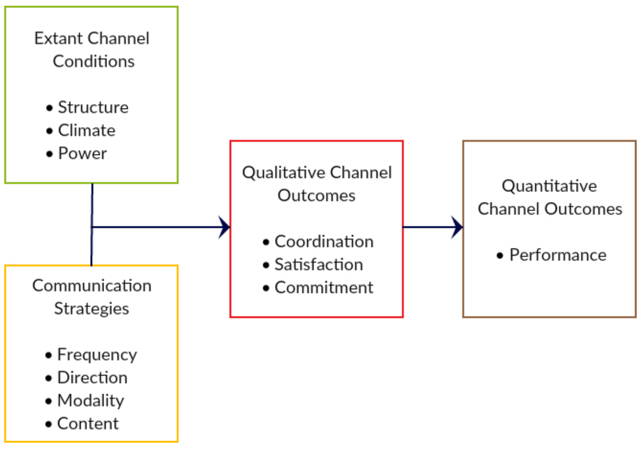


Figure 2.2 – Communication Model by Mohr and Nevin [68].

- **Power**: refers to the power conditions within the channel can be either symmetrical, with power balanced between parties, or asymmetrical, with a power imbalance [26]. In this case, power is related to the communication equality between interlocutors.

Communication Strategies

- **Frequency**: refers to the frequency and/or duration of contact between interlocutors. Though a minimal amount of contact is necessary to ensure adequate coordination, too much contact can overload interlocutors and have dysfunctional consequences [41];
- **Direction**: refers to the movement of communication within the organization hierarchy [35]. The literature discusses downward communication as flowing from the more powerful member to the weaker member and upward being the opposite. Within teams, this refers to the movement of communication between different roles.
- **Modality**: refers to the method used to transmit information. One straightforward way has been to categorize modality as face-to-face, written, telephone, or other modes. A second way has been to categorize according to the mode's ability to transmit "rich" information, or a variety of cues including feedback, facial cues, language variety, and personalization [64]. It can also be defined according to the formal/informal distinction. Formal modes are those perceived by interlocutors as regularized and structured, while informal modes are those perceived as more spontaneous and non-regularized.
- **Content**: refers to the message that is transmitted, i.e., what is said. Communication interaction can be analyzed for content by using pre-determined categories [2] or by asking the parties in an interaction what their perceptions of the nature of the content are [38]. Frazier and Summers [38] distinguished between direct (requests, recommendations, appeals to legal obligations, etc.) and indirect (exchange of information, discussions, etc.) the kinds of influence strategies within the content.

Channel Outcomes

- **Coordination**: refers to the synchronization of activities and flows by interlocutors [68].
- **Satisfaction**: refers to the affective or cognitive evaluation of the characteristics of the interlocutors' relationship [68].
- **Commitment**: refers to a behavioral component that reflects an allegiance with the interlocutors' relationship [92].
- **Performance**: can be assessed by considering several dimensions such as effectiveness, equity, productivity, and profitability [5].

3. RESEARCH METHODOLOGY

This chapter presents the research design and methods applied in this research, including the research question and goal, data collection and data analysis procedures.

3.1 Research Design

Singer, Sim and Lethbridge [85] argue that Software Engineering involves real people working in real environments. People create software, maintain and evolve it. Accordingly, to understand software engineering, one should study software engineers as they work. Consequently, it is natural to have some form of empirical investigation, which focuses on issues that matter.

Our research design uses and empirical and qualitative approach to identify a baseline for how the industry is using communication in DevOps. It is also characterized as an exploratory study, which allows researchers to define and discuss a problem which has not been studied previously on the topic under study [100]. We first conducted a snowballing literature review about DevOps and communication in order to gain knowledge on the area and to use as a baseline for the field study performed next (see Figure 3.1).

3.1.1 Phase 1: Literature Review

Literature Review started with *ad-hoc* searching on DevOps, using Google Scholar[1], in order to gain some knowledge on the matter. Later, we attempted to perform a Systematic Literature Review. However, we noticed that most information regarding DevOps originates from industry reports which has little scientific investigation and evidence. Consequently, we chose to perform a Snowballing Literature Review on DevOps and Communication, starting with a set of twenty articles found when using Google Scholar. Snowballing Literature Review is defined as a system for using the reference list of paper or the citations to the paper to identify additional papers [97].

In addition, during the execution of the field study, we conducted a follow-up on the articles published on DevOps. This monitoring was done in several databases so that it was possible to enrich the knowledge about the subject with the most recent studies. The search string used in both review was as follow: *("DevOps" or "Communication in DevOps")* for the DevOps subject and *("Communication in Software Engineering" or "Communication theory")*

[1] https://scholar.google.com

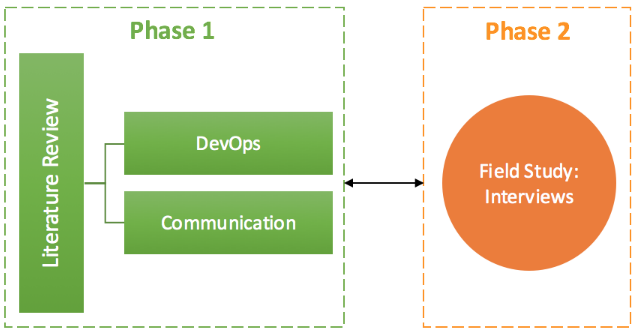


Figure 3.1 – Research design

for the communication one. The databases used were: IEEEXplore[2], ACM Digital Library[3], Springer Link[4], and Science Direct[5].

3.1.2 Phase 2: Field Study

Singer, Sim, and Lethbridge [85] explain that Field Study is a research method based on a recognition that software engineering is fundamentally a human activity, it is used to understand different aspects of real world environments. They complement that, to conduct a field study, it is necessary to use data collection techniques such as interviews, surveys, brainstorming, or notes. Adding to that, the most common used data technique is the use of interviews, also selected for this research, because it fits better when trying to understand general information on specific processes and personal knowledge.

Data Collection

In our field study we gathered data on communication in DevOps from the DevOps practitioners' point of view. To accomplish this, we conducted semi-structured interviews with the interview script consisting of a set of seventeen open questions. We chose to write open questions to explore the participant's individuality and to allow the interviewer to complement the interview if necessary. Once a draft of the interview was ready, we validated it through several rounds of content validation with a researcher experienced in empirical studies. Next, we piloted the interview script with a DevOps practitioner volunteer so that we could analyze if the questions were in agreement with the information we sought. The final version of the interview script can be found at Appendix A.

[2]www.ieeexplore.ieee.org
[3]http://dl.acm.org/dl.cfm
[4]http://link.springer.com/
[5]http://www.sciencedirect.com

For the selection of candidate participants, a preliminary search for possible candidates was held among the participants of DevOpsDays Oslo [23]. In addition, candidates were also sought through personal contacts, a set of DevOps groups' list, and other DevOpsDays events held in Porto Alegre (Brazil), and Trondheim and Oslo (Norway). The choice of these events and locations was due to the availability of the interviewers to travel and participate in those events to gather more people to participate in the research. After that, the technique of snowball sampling [61] was used, which involves asking people who have participated in a survey to nominate other people they believe would be willing to take part in the research as well.

In total, we interviewed 25 professionals. Each interview lasted in average 30 minutes, being the longest 1 hour and 7 minutes and the shortest 32 minutes long. The interviews were conducted via a virtual tool that allowed face-to-face communication and it was voice-recorded (with permission). The interviews were performed either in English or Portuguese during a period of six months and then manually transcribed to the participant's native language. We realized we had reached saturation when the information began to repeat itself, forming patterns of response, both in the interviews and in the feedbacks collected in the attended events. That is when we decided to stop the interview process and start the compilation of the findings.

Data Analysis

With the assistance of the MaxQDA[6] tool (student license), all information transcribed was analyzed via cycles of coding, thematic analysis and member checking reviews, see Figure 3.2 for a complete overview of the process. For the coding process, we decided to keep all transcriptions in the participants' native language considering the richness of the details provided by each. Nonetheless, all the coding was performed directly in English. Also, we employed a deductive [66] and inductive [14] coding technique. Then, we validated the code findings with another researcher (one of the member checking approaches performed) to seek the completeness of the conclusions. The purpose was to have another researcher double-checking the codes and data to tag the keywords, phrases, and paragraphs, and group them in short segments of data sets and useful constructs. It is important to highlight, that all researchers involved in the analysis and validation of the codings and results were fluent in Portuguese and English, being able to work with both languages of the transcriptions and codifications.

While data was still being collected, an early data analysis was initiated to help with improvements on the data-gathering cycle – to provide insights on which questions of the interview script to emphasize to get more data. Cruzes and Dybå [17] define Thematic Analysis as a method for identifying, analyzing, and reporting patterns (themes) within data.

[6]www.maxqda.com

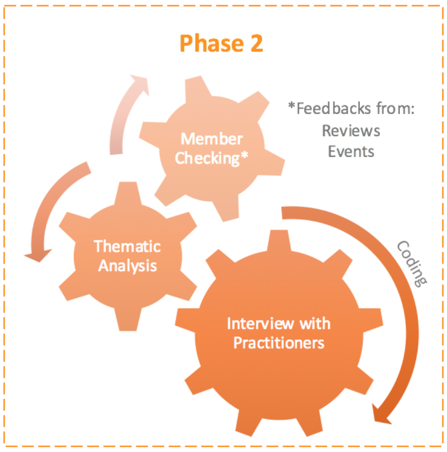


Figure 3.2 – Phase 2 - Field study data collection and analysis process

Saying that it also minimally organizes and describes the data set in rich detail and frequently interprets various aspects of the research topic. Figure 3.3 presents the suggested steps that were performed when conducting the thematic analysis.

After performing the initial reading of the data, we started identifying specific segments of text and then labeling it into codes and themes. For coding, Cruzes and Dybå [17] best define the deductive and inductive techniques as follow:

- The deductive approach allows the researcher to define a structure of initial codes before the actual coding of the data. These preliminary codes can help researchers integrate concepts already known in the literature. In this study, the 'start list' of codes came from the the concepts of the communication model presented by Mohr and Nevin [68] (e.g.: structure, climate, power, frequency, direction, etc.). So, while reviewing the transcripts, we would code specific segments of the text into one of those major codes to be later assigned to specific themes.
- The inductive approach limits researchers from erroneously 'forcing' a preconceived result. During the inductive approach, data are reviewed line by line in detail, and as a concept becomes apparent, a new theme is assigned. Upon further review of data, the researcher continues to assign themes that reflect the emerging concepts, highlighting

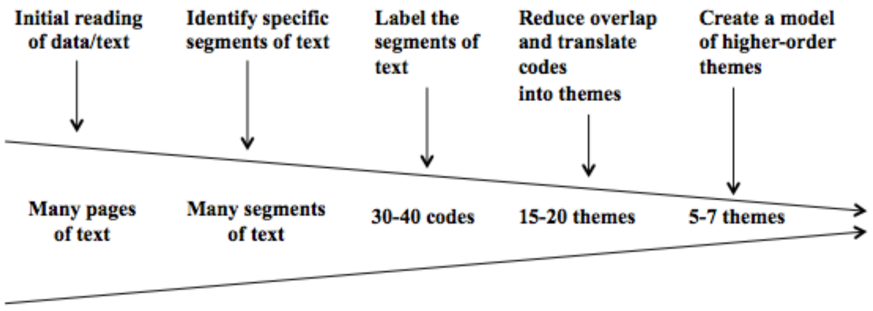

Initial reading of data/text
Identify specific segments of text
Label the segments of text
Reduce overlap and translate codes into themes
Create a model of higher-order themes
Many pages of text
Many segments of text
30-40 codes
15-20 themes
5-7 themes

Figure 3.3 – Thematic analysis steps by Cruzes and Dybå [17]

and coding lines, paragraphs, or segments that illustrate the chosen concept. In this study, this technique was applied to gather new insights within each concept of the communication model – e.g.: within direction it was later created "*from Devs*", "*from Ops*", "*from management*", "*bidirectional*" and so on.

The results were used to guide the discussion and validation of the findings with the participants of the study and other active DevOps practitioners. At the validation session, incremental feedback was collected through member checking. This technique is particularly well suited to most studies of software engineering, getting feedback on the findings from the subjects who provided the data in the first place [80]. Two forms of member checking were applied: (i) coding and analysis review with co-advisor; and (ii) seeking DevOps professionals opinions through the participation at 4 large conferences in the course of 2016: DevOpsDay Oslo [23] (guest speaker), ICGSE [56] (presentation of a short paper [24]), ComputerWorld Oslo [13] (guest speaker), and AgileBrazil [6] (topic approached in an open space).

4. RESULTS

This chapter presents the participants profile, along with the results based on the communication model introduced in Chapter 2, grouped by the Channel Conditions, Communication Strategies, and Channel Outcomes. It will also present other findings found with the research, such as major factors for a successful communication and major challenges to successful communication. See Appendix B for a complete view of the results in the form of a mind map. For each finding presented in there, the number in parenthesis indicate how many people have approached that subject, counted by role (e.g.: "*relational / joint planning (19: 8D11O)*" meaning 19 participants, 8 Devs (D) and 11 Ops (O)).

4.1 Participants Profile

Most participants have at least one year of experience working with DevOps, one of the participants reported having more than ten years of experience working as a sysadmin that also assists the development team. When asked about their team's location, 15 participants said they work in distributed teams, six said that all team members worked on the same site, and four chose not to disclose that information. From the 25 participants: 14 are Brazilians, and 11 are Norwegian; 12 are working as developers and 13 as operations. When questioned about what DevOps team meant to them, three different answers were obtained: (i) Dev and Ops - same team: to represent a team that has developers and operations personnel working within the same team; (ii) DevOps team - shared skill set: to represent a team of DevOps professionals - those who have both development and operations skills, those who work in both roles; and (iii) Dev team and Ops team - close collaboration: to represent one development team and one operations team working together when necessary. Table 4.1 summarizes the background profile of the study participants.

4.2 Channel Conditions

Channel Conditions are related to the nature of exchange relationship between interlocutors (structure), to the feelings that the interlocutors have when exchanging information (climate), and about the power conditions in the communication (power). This section presents main results found on those subjects.

Table 4.1 – Background profile of the study participants

Participant	Nationality	Role	Yrs of exp. DevOps	Team Distribution	DevOps Team Structure
P1	NOR	OPS	5	-	Devs and Ops - same team
P2	NOR	DEV	-	Distributed	DevOps team - shared skill set
P3	NOR	DEV	4	Distributed	Devs and Ops - same team
P4	NOR	DEV	2	Same Site	Dev team and Ops team - close collaboration
P5	NOR	OPS	4	Same Site	Devs and Ops - same team
P6	BR	DEV	1	Distributed	DevOps team - shared skill set
P7	BR	DEV	5	Distributed	Dev team and Ops team - close collaboration
P8	NOR	OPS	6	Same Site	Devs and Ops - same team
P9	BR	OPS	3	Distributed	Dev team and Ops team - close collaboration
P10	BR	DEV	2	Distributed	DevOps team - shared skill set
P11	BR	DEV	4	Distributed	Devs and Ops - same team
P12	BR	OPS	2	Distributed	DevOps team - shared skill set
P13	BR	OPS	5	Distributed	Dev team and Ops team - close collaboration
P14	BR	DEV	2	Distributed	Dev team and Ops team - close collaboration
P15	BR	DEV	1	Distributed	Dev team and Ops team - close collaboration
P16	BR	OPS	6	Distributed	Dev team and Ops team - close collaboration
P17	BR	DEV	2	Same Site	Dev team and Ops team - close collaboration
P18	NOR	OPS	-	-	Devs and Ops - same team
P19	BR	DEV	4	Distributed	DevOps team - shared skill set
P20	NOR	OPS	-	-	Devs and Ops - same team
P21	BR	OPS	4	Same Site	Devs and Ops - same team
P22	NOR	OPS	-	-	Dev team and Ops team - close collaboration
P23	NOR	DEV	10	Distributed	Dev team and Ops team - close collaboration
P24	NOR	OPS	4	Distributed	Devs and Ops - same team
P25	BR	OPS	6	Same Site	Dev team and Ops team - close collaboration

4.2.1 Structure

Structure condition impacts directly on the way people coordinate tasks and how their teams perform. Participants reported both ways of structuring their communication: relational/joint planning and discrete/short planning. Even though, the two roles contributed with examples on this subject, during interviews we noticed that operations personnel were more open about this topic, willing to share more examples in this end. Figure 4.1 presents an overview of the results.

The relational manner happens through formal meetings where they discuss the planning for the next release or any other issue that might be affecting them. The main meetings and processes mentioned were: pairing, scrum meetings (backlog review, retrospective, planning, etc.), SAFE meetings, and tech talks (showcases). Participants P12 and P7 exemplified this:

> "*In our team we have a retrospective meeting and have a 'TechOps' meeting once a month, which is called Tech Talk, or something like that, which is a time when we share technological topics that we are doing, working, or has been investigating.*" (P12).

> "*We use SAFE, which is an agile implementation for enterprise companies. We have the majority of programs and projects delivering service in a 3-month cycle.*

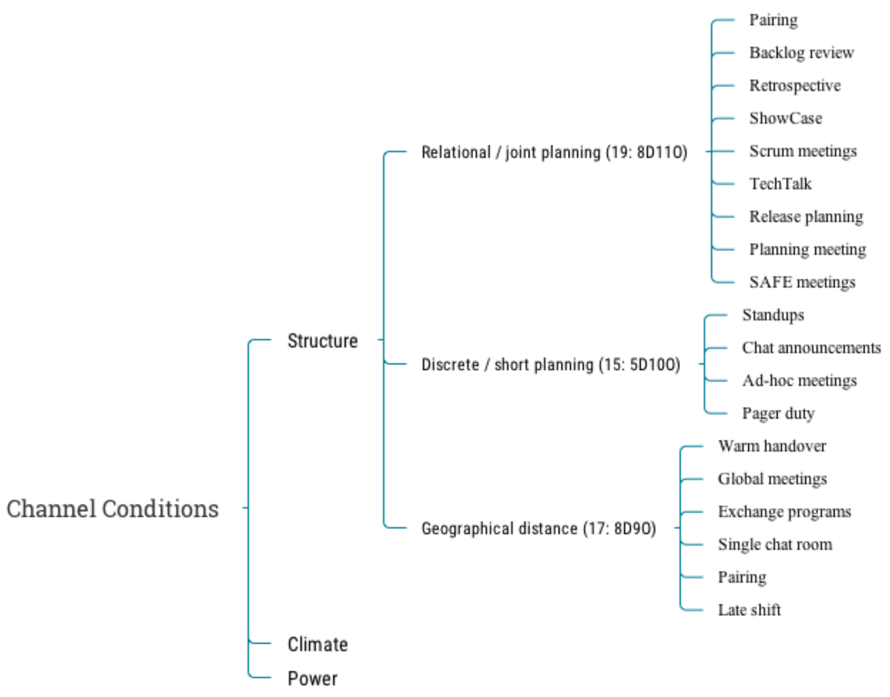


Figure 4.1 – Results: channel conditions - structure

This is our release cycles. During these intervals, there are planning ceremonies for releases cycles, and there are grooming meetings, where we are always actively participating. In these meetings we also raise barriers, dependencies between teams." (P7).

The discrete manner happens every day on their communication through stand-up meetings, chat announcements, *ad-hoc* meetings, and pager-duty calls. Participants P1 and P9 exemplified:

"*Usually during stand-ups, we have some of the upcoming updates. We usually add small things more often, so, the new feature is going to be still compatible. It's an effort for everyone to know about feature coming soon but it's very seldom that the feature is so groundbreaking and everyone should take into account code changes or something.*" (P1).

"*We use a lot of HipChat room to report. Any team member can place info there like "went to use this application and was not in the air" or "application X feature is not opening," that kind of thing. Then, someone on the team who has an overview of the problem will tell who should work on that and provide feedback on the chat.*" (P9).

When a team is geographically dispersed, they usually add to that list the following procedures to keep the communication flowing with all team members as P9 and P6 later exemplified: warm handover, global meetings, exchange programs, single chat rooms, late shift.

> "*Well, we usually do a handover, because of the time zones, we currently have 3 different time zones, so we do a warm handover like: "I've been doing this, the team has to keep doing that" or "I've done such a thing, so you need to keep an eye on it." and so on.*" (P9).

> "*We have global meetings with all members, including the remote ones. Sometimes the time zone issue was a little challenging. We had to find a way and time for everyone to participate, but the rule is that all team members are present.*" (P6).

4.2.2 Climate

Communication climate is associated with what feelings/aspects impacts the most on communication. The interviewees listed 19 factors that contribute to that. The most relevant being: working together as a team (fostering teamwork), trusting each other work, and having a good technical knowledge. Figure 4.2 presents an overview of the results.

Working together as a team (fostering teamwork) was approached in the sense of mutual support. As a team member, the ability and desire to work cooperatively with others on a team; as a team leader, the ability to demonstrate interest, skill, and success in getting groups to learn to work together [15]. P13 and P23 mentioned the following:

> "*For day-to-day problems, we always try to engage new hires for them to learn and come to ask for help, and already have this culture of coming and talking directly to the person without being afraid to solve the problem.*" (P13)

> "*It's not just about the operations team and us, once things are not working properly in any environment, then the error could be on many different bases, developers and operations team will work together to solve this when it comes to the application that we work on.*" (P23)

Even though all participants agree that trust is important for communication, only participant P1 reported that currently there is no trust among the colleagues, and this has negatively affected their satisfaction towards their peers:

> "*We currently have a big gap between Dev and Ops where developers are very much into writing code as well as running it. But operations team, on the other*

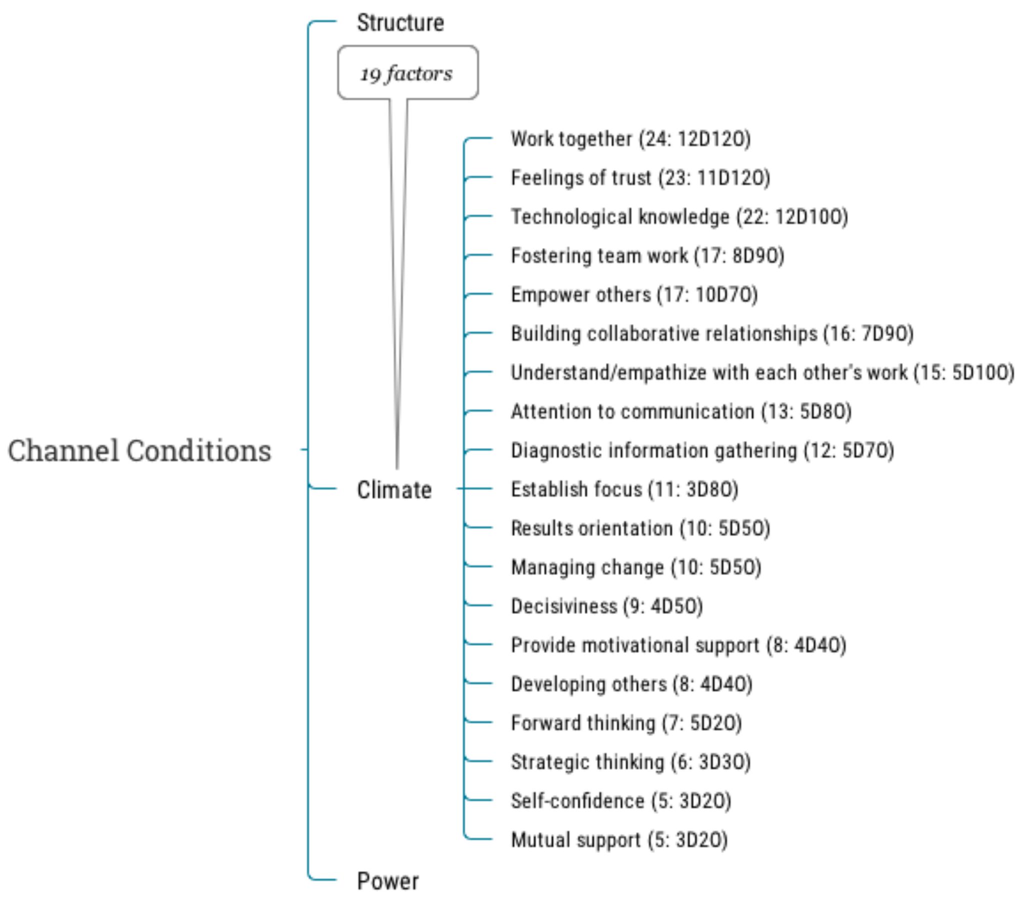

Channel Conditions
Structure
19 factors
Climate
Power
Work together (24: 12D12O)
Feelings of trust (23: 11D12O)
Technological knowledge (22: 12D10O)
Fostering team work (17: 8D9O)
Empower others (17: 10D7O)
Building collaborative relationships (16: 7D9O)
Understand/empathize with each other's work (15: 5D10O)
Attention to communication (13: 5D8O)
Diagnostic information gathering (12: 5D7O)
Establish focus (11: 3D8O)
Results orientation (10: 5D5O)
Managing change (10: 5D5O)
Decisiviness (9: 4D5O)
Provide motivational support (8: 4D4O)
Developing others (8: 4D4O)
Forward thinking (7: 5D2O)
Strategic thinking (6: 3D3O)
Self-confidence (5: 3D2O)
Mutual support (5: 3D2O)

Figure 4.2 – Results: channel conditions - climate

> *side, does not want to let developers have insights on how code is running because they are responsible for uptime and first race. So, no, there is not enough trust, of that I'm sure.*"

P1 also complemented by saying that:

> "*There's a big uncertainty that this code will end up this way and not in another way. Sometimes, Ops change files simply because they think this is more secure, but it will simply not work this way. This problem is a lack of trust.*".

All the other participants reported having a good and trusting relationship between the teams as mentioned by P8:

> "*When I think about trust, you can't have effective communication unless you trust each other - you can't tell people bad news - you can't have the conversation that you need to have in order to be effective if you don't trust each other.*"

An interesting detail highlighted by participant P4 is that trust is not only related to personal feelings but with the sense of team, to have a shared goal:

> "*I think there's trust. Most people have the opinion that we have to achieve a goal together. So, it's not related to feelings or anything, but with to get the work done.*"

Good technical knowledge was also approached in the sense of mutual support and the ability to demonstrate a depth of knowledge and skill in a technical area. P3 and P21 expressed this by saying:

> "*If I'm quite confident of these changes, and I know what I'm doing, I'll probably just do it myself. I'll just tell them what I'm doing. If it's something that I'm not sure about, them, I'll just ask them for either showing me how to do it or either just doing it for me. But it just depends on the context.*" (P3).

> "*Usually we have a default of two developers per review. So sometimes they are developers who do not necessarily have the infrastructure knowledge. So we sit down together and make a kind of pair review, one explaining what each part of the code is doing to and the other how the architecture/infra works.*" (P21).

It's worth noting that while most developers mentioned "empower others" as an important factor, most operations personnel talked about "understanding and empathizing with each other's work." This indicates that even though developers seek more autonomy, they tend not to think that things might work that way because of a business decision and not because operations do not want to cooperate. P19 and P21 talked about that:

> "*I think one thing is what kind of access the Dev has and what kind of access the Ops has. Once this information is clear, for example, logs, if my developer has access to the logs, just like the Ops does; if both of them can do an ssh on the server, or do a remote access and look at those logs, this is not a problem, it turns out to be a natural thing, it's about demand, if one needs information, go and get it.*" (P19).

> "*I feel like this lacks a bit, this thing of coming from the other side, the development coming to know a little more about the Ops. They usually only come when they have some problem or need some new solution.*" (P21).

4.2.3 Power

Participants reported that power of decision and a more active voice are highly dependent on the situation and the autonomy of the team. It is interesting to point out that when questioned about this type of situations, the participants took longer to answer. This

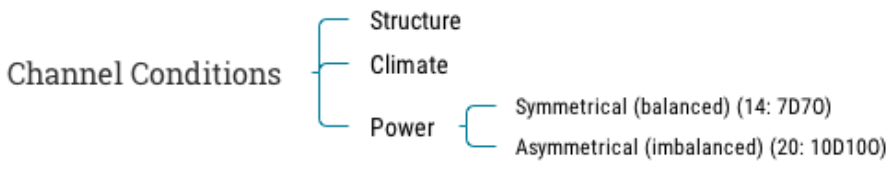

Channel Conditions
Structure
Climate
Power
Symmetrical (balanced) (14: 7D7O)
Asymmetrical (imbalanced) (20: 10D10O)

Figure 4.3 – Results: channel conditions - power

question allowed them to think about the other side as well (the ones in opposite roles). It encouraged them to put themselves in the other's shoes, noting that this was more situational, where, depending on the context, one might have more decision power over the other and that this was fine for most of them. Figure 4.3 shows the balance of their responses, and the following examples were mentioned:

- Symmetrical power:

  > "*To optimize things, it's the Ops. So, when something hits performance, Ops can get to Dev and ask to sort things out. So Ops have a lot of active voice regarding settings for the applications to be running properly. The Dev has the most active voice in deciding how things are going to be implemented. For example, we say we have to have a button to make things easier, and Dev says that this is not necessary.*" (P13).

  > "*It kinda depends on how what's a given a choice will affect the most, if it affects how Ops will work, then operations will have the most powerful voice, and if it affects mostly how development process is going to be, it's usually development that has the last saying. So, actually, I think that the power distribution is kinda even.*" (P4).

- Asymmetrical power:

  > "*Decision-making power, neither Ops or Dev. If I could scale priorities, the power of argument, the Ops were always at the bottom, and then the Dev on top, and first, the business. Business is always the boss.*" (P16).

  > "*Sometimes the developer needs to add a feature, but he can not explain it to the Ops. Then, an endless discussion about the need arises and whether it is possible to do so. I've also seen conversations about needing to install something and Ops saying it would not be possible. So, the developer had to change the strategy because he wouldn't be able to do that and think of other alternatives*" (P14).

4.3 Communication Strategies

Communication Strategies are related to the use of a combination of communication facets (frequency, direction, modality, and content) [68] to achieve a specific goal, to send a specific message. This section presents main results found on each facet.

4.3.1 Frequency

The participants reported that the frequency of the communication varies according to the subject being discussed (if it is urgent or not), how the team is organized, and the availability of other co-workers. The frequency of the communication was divided into the following time slots: daily, weekly, monthly, annual, and *ad-hoc*. Figure 4.4 presents an overview of the results.

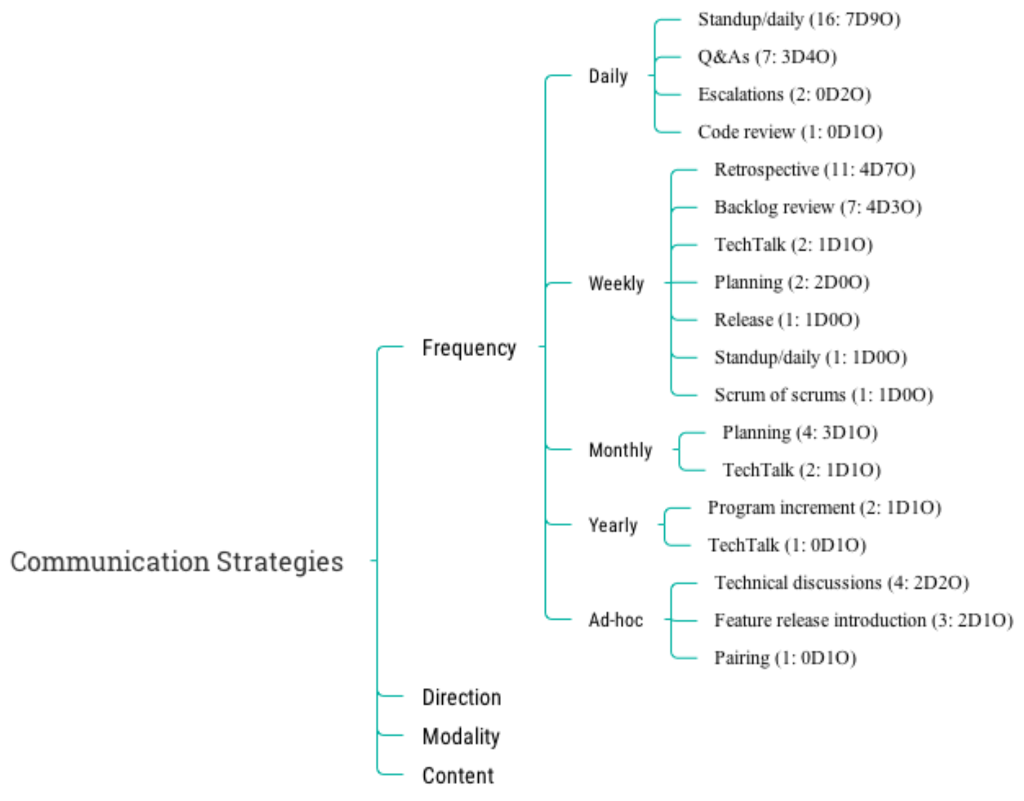


Figure 4.4 – Results: communication strategies - frequency

Most of the participants stated that their daily communication happens through stand-up meetings and Q&A sessions (face-to-face or via chat) as P20, P22, and P2 describe:

"*We have daily scrum meetings in the morning. We have a Flowdock or Hangouts channel, and we communicate as more questions arise and so on.*" (P20).

"*We talk continuously throughout the day and night even. We try to keep the chat channels very operational, which means that people, whenever they are awake, they are in the channel speaking, either about their personal life or work related.*" (P22).

"*We have war room IRC channel, so when there's an emergency or something breaks, then everyone get into that room, and we start coordinating.*" (P2).

Other meetings that were most mentioned were retrospectives and backlog reviews. The participants pointed out that these happen on a weekly frequency. However, it can be directly related to the duration of the sprints of each team or the frequency of their releases. P24 and P3 commented the following:

"*We have a weekly incident review to talk about what kind of alerts they got last week, which of these alerts should be better so we could have a playbook for people to figure out what to do, if we could have automated this thing, could one as the operations team do something to operate this thing, log rotation, can anything be done to improve that process, if everyone that needs to have access to things actually have them, especially outside our atmosphere.*" (P24).

"*We have these weekly video meetings which we go through this list of tasks and prioritize them. Sometimes, more often, if required.*" (P3).

A related point to consider is that the frequency of the communication normally happens synchronously (instantaneous), even in distributed teams. The only examples where it happens asynchronously is when it involves escalations or e-mail notifications as mentioned by P13, P2, P14, and P18:

"*Escalating the ticket is more difficult, that's when the problem is very complicated. Then, we work with the Devs and pass on the information they give us.*" (P13).

"*So, now we created this mailing lists and make sure that there's always, whenever there's a maintenance planned or there was some accident that we always have a person in the e-mail and the e-mail will be like this: this is the time, when, person responsible, next update, etc. So, we improved our communication a bit based on that.*" (P2).

"*In the current project, we have used a tool that has worked fine so far, which is a weekly e-mail, in the newsletter idea, only to have a review of what has been done when we have many stakeholders.*" (P14).

"*Communication via e-mail takes more time, because it takes time to compose an e-mail, to get it right, even though voice communication is quick and easy, it still takes time because one can't concentrate on something else.*" (P18).

4.3.2 Direction

The direction is associated with the communication movement, with who usually starts the conversation and brings more information to the team. Just like in Power, most participants agree that the movement is situational, but mostly dynamically (bidirectional). Figure 4.5 presents an overview of the results.

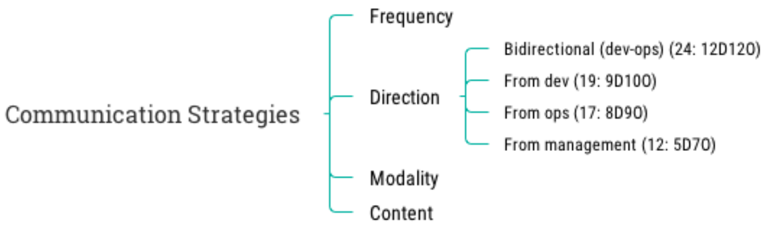


Figure 4.5 – Results: communication strategies - direction

Direction can also be related to the type of information that is being exchanged (content). For example, bidirectional communication usually happens when they need to solve a problem or plan together. P20 and P10 exemplified:

"*I think it depends on the phase of the application. In the development phase, before it goes live, I think most of the communication starts with developers. After we are in the operational phase, it's usually the operations that will start interaction with developers. But I feel working in one team has more balance than it had before, because of the back and forth.*" (P20).

"*The sysadmin helps us with this part of understanding the infrastructure, permissions, etc. And we believe that developers contribute to best programming practices, agile approaches. This is the information that we exchange.*" (P10).

The communication that comes from the developers sounds more inquisitive, in the sense of they trying to understand better how an application or service behaves. Sometimes, developers engage the operations personnel in their routines to assist in some tests that have to be performed manually, or in some maintenance alert from the Dev side as well. P13 and P7 shared some good examples:

> "*Devs usually come to us asking for help to run some tests, so sometimes we help them by running the tests in some applications, in a very informal way, like, get the cell phone and testing it.*" (P13).

> "*When we do maintenance, we always get in touch with them, so they're always aware of what's going on. Like, "Man, we're going to do maintenance on service X. Then, from noon to 4 pm you will see warnings in the monitoring system, and you can ignore it." that kind of thing.*" (P7).

From the operations side, information is more informative: server status, how the application is behaving with the changes made by the Devs, etc. Ops also tend to share insights on code/service optimization, just as P5 and P23 mentioned:

> "*So, I ask about applications and how that works. If I see something that is not working opting out, then I'd ask how that works and then saying "ok, because that thing is causing some problems in X. So I think maybe you could change it to be more like Y." that kind of stuff.*" (P5).

> "*If it's something urgent happening in production, I think that Ops people monitor the applications. If something bad happens, they contact the technical responsible; this person will then contact the developers. We do have a say on how things are going to be implemented or how should things run in prod environment, but we're mostly using a more quiet approach, using a technology that doesn't need that much input from us.*" (P23).

The communication that comes from the managers is more decisive. They usually define the tasks to be performed by the teams and review the approval of the releases. P5 and P16 commented the following:

> "*We have a PO [Product Owner] that usually does a great job in planning for us, at least telling us which feature to develop first.*" (P5).

> "*Usually the approval comes from the test together with the PM [Project Manager], along with the business, they would do another meeting before the "Go" or "No-Go". The tester had the power to say if in the perspective of quality the system the system is appropriate and with all the features. Then, the business has the power to decide whether the company would really want to deliver this or not.*" (P16).

4.3.3 Modality

There are two ways of transmitting information: one by using informal methods, and the other via formal ones. According to Mohr and Nevin [68], informal modes are those perceived as more spontaneous and non-regularized. They usually are more personalized, spontaneous, and can occur outside the organizational chart or premises. While formal modes are those perceived by organizational members as regularized and structured. It refers to communication that flows through written mode or formal meetings. Figure 4.6 presents an overview of the results.

The most commonly used means of communication are chats and face-to-face conversations. Participants liked these better because they allow them to get answers faster by using a much simpler approach. P14, P10, and P22 reported the following:

> "*We use the chat a lot during the day. Mostly to inform what is happening, informing that, for example, "hey, I am updating this part of the infrastructure." So, it's important for someone who is working on that to know what's going on.*" (P14).

> "*Today in the project that I am, we do not make any more requests for the Ops team because we have a Ops person inside the team. So we do not even open the ticket, we look at the guy and say "hey, give me the permission there." then he gives permission to us.*" (P10).

> "*Of course the operations team is back to back here. So, they can just go around and talk to people. In essential, most of the developers live in the same hallway. So, of course, we talk over lunch, and there are lots of things that happen in those talks.*" (P22).

As a formal mode of communication, formal meetings (with all those Scrum or SAFE meetings), e-mails (being the last form of communication used due to its formality),

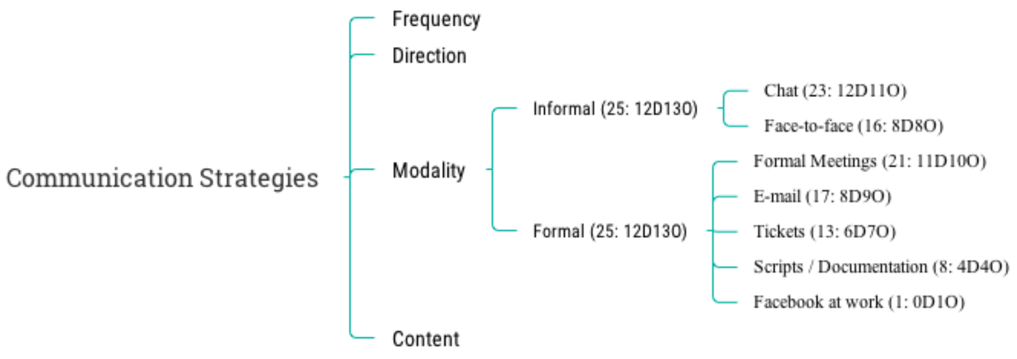


Figure 4.6 – Results: communication strategies - modality

and tickets (specific tools to manage tasks and priorities) were the most emphasized by the participants. P8, P5, and P23 elaborated on this:

> "*We followed the scrum process. So, we'd do sprint planning, daily stand-ups, retrospectives, demos, and on top of that, we'd have to, as a tech lead, to do the planning and reporting to management teams.*" (P8).

> "*Very little e-mail, but few things we send by e-mail. Things that are security and stuff, they usually go in an e-mail. Communication with external parties, they usually go in an e-mail.*" (P5).

> "*We use JIRA for getting things into production, so to do that, you raise a ticket to the operations team so that they can manage the deployment.*" (P23).

Tools

When talking about ways of transmitting information (the method used), participants mentioned the tools they normally use, briefly discussing positive and negative points presented in each. Chat tools like Slack, Flowdock, and HipChat were the most cited due to their easy usability; availability of both group and individual conversations; and the 'bots' (automation of tasks) they offer to make it easier for teams to have an awareness of things that are happening in their projects. In addition to these, Google Hangouts, Skype, and Team Viewer were the most cited for when conducting video conferences and pairing due to the quality of the service offered. Other communication tools mentioned were: Microsoft Link, IRC, IM, HP Exstream Cloud, Google Plus, Skype for Business, HP My Rooms, Let'sChat, XMTP, and WhatsApp.

Technological tools mentioned to help through the day were: AWS, SaltStack, TFS, Wiki, PagerDuty, Trello, JIRA, Chef, CloudStack, GitHub, Bitbucket, Docker, Office365, OneNote, AlwaysTicket, Rally, Confluence , Track, GIT, Cibola, Genix, Puppet, Nexus, VersionOne, Splunk, and Jenkins.

4.3.4 Content

Content is associated with the message that is transmitted or what is being said. It has two categories: indirect and direct. Mohr and Nevin [68] explained each as: indirect communication is designed to change the target's beliefs and attitudes about the desirability of the intended behavior, no specific action is requested directly; and direct communication strategies are designed to change behaviors of the target by implying or requesting the specific action that the source wants the target to take. Figure 4.7 presents an overview of the results.

From the examples mentioned by the participants, the following sub-categories were created within indirect communication: exchange of information, which relates to awareness; discussions, which relates to talks in order to reach a decision or to exchange ideas; Q&As, which relates to daily questions and answers about things that impact their daily work; and not work related, which relates to a more personal conversation. P15, P21, P18, and P17 shared the following examples on each sub-category:

> "*We use Slack a lot, so we have channels for each subject for each environment, we try to use the one that comes closest, for example, with a production problem, we enter the production channel and report the problem there, then, someone will act.*" (P15).

> "*At times, there are even cyclical discussions, there is a subject related to the project, for example, we use microservices, then there's a back and forth discussion about the best way to do something in micro services. This is a discussion that we have had about 5-6 times and never comes to an end and then always ends up coming back, or a person finds new information and ends up reopening the discussion.*" (P21).

> "*Generally, we will receive IT, technology related requests to fulfill some needs. There's something that started now called "Core Programming," so straightforward, and they want technical things like "How to I filter something? How do I handle arduino? How to I use components?". Generally, it's always something IT related, if the person can find the answer themselves, then they do. Otherwise, they just raise a request for us to do that. But again, it's anything IT related.*" (P18).

> "*We are all buddies here, we go out together and get along really well, it's really cool. This is not just the technical team, but the entire company. It's good to have this kind of involvement outside the work environment.*" (P17).

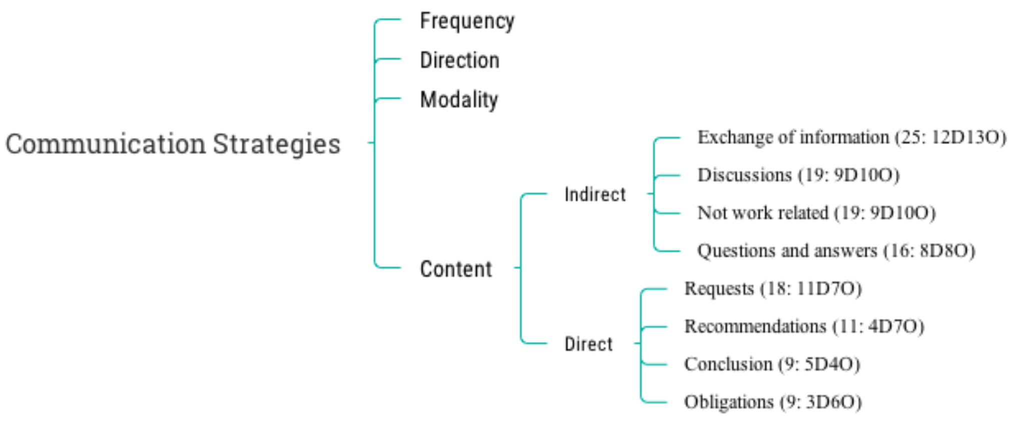


Figure 4.7 – Results: communication strategies - content

For the direct communication, the following sub-categories were created: requests; recommendations; conclusions; and obligations. P6, P5, P21, and P20 exemplified with the following:

> "*When he [Ops] sees something that is not functioning properly, he triggers an engineer to solve the problem immediately, to work on that task, or he already creates a defect, defines that it is urgent and has to be solved within that sprint because that can lead to a lot of problems for teams using that platform.*" (P6).

> "*I also advise the developers a lot on how they should make their software easier to operate and they ask me for advice. Also, when they have a bug in their production software, they ask me advice on what the reason is and how or what we could do to prevent it from happening.*" (P5).

> "*I just keep things on e-mail that I think need to be logged for some reason or because the final decision needs to be kept. We often come in, get together, and discuss everything, and then we e-mail the whole team just to formalize.*" (P21).

> "*Like, there will be some kind of "guard duty", someone that is responsible for it at that specific time and that should respond to it, or ship it to somebody else that has time available to fix it.*" (P20).

4.4 Channel Outcomes

Channel outcomes are related to how the channel conditions and communication strategies impact on the outcomes of the team. The investigated channel outcomes were coordination, satisfaction, commitment, and performance. This section presents the main results found organized by each of the outcomes.

4.4.1 Coordination

Coordination refers to the integration of the different parts of the organization to accomplish a collective set of tasks [68]. Participants reported to use chats and other tools like Trello and JIRA to coordinate their tasks, and that it works perfectly for their business need. They also said that those tools allow everyone involved on the project to have a better overview of the same. P3 and P6 shared the following examples:

> "*They use group chats when they had to do a production delivery, to deploy an application, they made a linking group chat where operations and all the other teams were involved.*" (P3).

> "*So what we usually do is, the Scrum Master first identifies whether the information we use to track things is up to date, for example, whether the person working on the defect or story is the most appropriate one or not, if the data is updated and such. This is done on our dailies and is important so that all information regarding the project is the most up-to-date in case someone needs it.*" (P6).

4.4.2 Satisfaction

Satisfaction refers to either the effective evaluation or cognitive evaluation of the characteristics of the channel relationship [68]. Most participants said they were satisfied with the means of communication used. However, they also pointed out some aspects that concern them, for example, too much noise in the communication channels. Many e-mails are sent daily, and in the middle of so many messages to read, something important can be lost in there. They feel the same with chats. The lack of useful information is also something that troubles them. P14 and P1 elaborate a little more on this:

> "*I think the negative impact is that it generates a lot of noise in the communication, a lot of noise, so I see a lot of people having a hard time following the conversations when we were in that environment with the chat very active. We had several complaints of "I can not keep up with the chat." We often had many conversations that were between 2-3 people, only that sometimes they had 3-4 simultaneous conversations of different people. Then, yes, these things really get noisy.*" (P14).

> "*So, operations currently does not provide us metrics or statistics. So, we have to dive for this information ourselves. Kinda trying to figure out which metrics make sense for us now. So, this is like a big cold 'bloom' with operations and this company. They are not providing us with tools which we should have. Even typical things like loads for different servers, we have them sometimes, other times, systems are down, and no one knows about.*" (P1).

4.4.3 Commitment

Commitment implies a behavioral component that reflects an allegiance to a channel relationship [68]. It was pointed out by the interviewees in the sense of commitment to the tasks to be accomplished and information to be shared. P23 and P13 said the following:

> "*It's not just about us and the operations team, once things are not working properly in any environment, then the error could be on many different bases, devel-*

opers and operations team will work together to solve this when it comes to the application that we work on. It might take some time to find the right place to address the problem." (P23).

"*There is a communication that is made in a tool that we have that is Facebook-At-Work. It is an application of the company. It is just like Facebook, only private to the company and filled with things that we are working on. The idea of the company is to show what you are doing all the time, to be clear to everyone, what people are doing, you have to publish it there. The Dev will say "people, I finished the project X, and I hope that next week you're going to be using it!", it is just like a Facebook post.*" (P13).

4.4.4 Performance

Performance is a multidimensional outcome measure that can be assessed by considering several dimensions including effectiveness, equity, productivity (efficiency), and profitability [68]. In this regard, the participants focused more on examples where factors ended up impacting the productivity of the team. Some of the factors were: excessive meetings, differences in the priority of tasks to be performed, lack of information and autonomy. P12, P18, P14 and P19 elaborated a little more on the following:

"*It was due to their priority, they had an urgency in another project, and they had to allocate these people in this project. We did not communicate for two days. Once we were able to get everybody on the same page again, they had another change in priorities and were out for another day. So getting the context of what was being worked up was much more difficult.*" (P12).

"*The team that we used to have, we used to have too many meetings. Say weekly meeting or once a day for status. So, with the time that one spent preparing for the meeting, one didn't get any work done, which is unfortunate. In the opposite it's also true that if one doesn't have any meetings, one is just crunching, crunching, crunching, and end up actually saying that you've crossed each other.*" (P18).

"*Sometimes, I think the main thing is lack of communication, this impacts a lot. What I usually see is teams that are not into continuous communication, if they are distributed teams, the lack of communication in the chat is very more worrisome than the noise.*" (P14).

"*I think it makes a lot easier, that one does not create a dependency on other people. One has the autonomy to go to the source of information, you know how,* *where to look, and how to use that information. It is worse if one has to count on someone who may not be on the team to do things one needs.*" (P19).

4.5 Other Findings

Other findings are related to other facts that were also addressed during the interview but are not directly linked with the communication model mapped here. Nonetheless, it still contributes to a better understanding of how communication happens in DevOps. This section presents the major factors and major challenges for successful communication in DevOps.

4.5.1 Major Factors for Communication Success

When asked about the major success factors for communication in DevOps, 12 factors were mentioned by the participants (Figure 4.8 shows the complete list). During analysis, most of the 12 factors were grouped into three broad categories, namely: to work together, to have empathy towards each other, and to be clear and objective in the communication.

Working together was approached in the sense of having everyone sharing the same goal, knowledge, and autonomy. Also, working together to coordinate tasks by getting everyone involved/informed, and once decisions are made, commit to the work. P8, P2, P14, and P7 commented:

"*So, the major factor for communication success is to get everyone on the same team to deliver software, because once everyone is on the same team, with the*

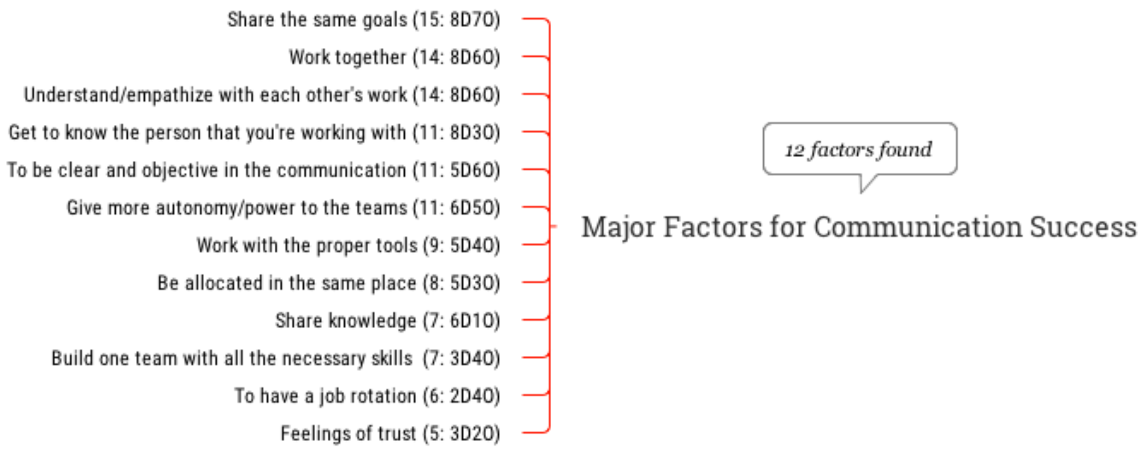

Share the same goals (15: 8D70)
Work together (14: 8D60)
Understand/empathize with each other's work (14: 8D60)
Get to know the person that you're working with (11: 8D30)
To be clear and objective in the communication (11: 5D60)
Give more autonomy/power to the teams (11: 6D50)
Work with the proper tools (9: 5D40)
Be allocated in the same place (8: 5D30)
Share knowledge (7: 6D10)
Build one team with all the necessary skills (7: 3D40)
To have a job rotation (6: 2D40)
Feelings of trust (5: 3D20)
12 factors found
Major Factors for Communication Success

Figure 4.8 – Results: other findings - major factors for communication success

same priorities and the same goals, all other problems are much, much easier - the technical problem becomes easier to solve, the cultural barriers become easier to solve." (P8).

"*To me, that why we succeeded. When people saw the scripts I used to install the machines, they were curious, and I said I used this tool, and if one uses it as well, one will be done quickly. Then, it started to spread, because someone hears that it was possible to install machines so quickly. It had to be convenient for people to adopt this new way of working.*" (P2).

"*Success is not having people lost, it's all about knowing where we want to go and having a sense of what the next things are going to be. Having everybody knowing the important points of the project is a success factor for me.*" (P14).

"*To have the teams engaged, one need to pass the sense of ownership. When people own the product, their actions and posture are different. If people are not engaged in making things happen and feeling that they own the product, not just the code, then, nothing works.*" (P7).

To have empathy towards each other relates to better understanding each other's work and also getting to know the person that one is working with. This type of approach usually facilitates when exchanging favors or a request is required. It becomes easier since one already has a certain familiarity with the other person and one already understands when and how to approach them best. P24, P12, P3, and P4 exemplify:

"*Both need to pay more attention to the other side, we need to have the on-call team to check that developers have introduced a new feature and making sure that this runs the best way possible, but also having the developers team to look into the operations side of the feature before developing it.*" (P24).

"*This thing about letting one's guard down, like, knowing that the person has other priorities. That people's priorities might not be the same as ours. The knowledge that the person acquires is not the same as what one has acquired. It was very crucial, to go more calmly, to have more empathy.*" (P12).

"*I think it's part of the culture thing, really get to know them better, their reasoning and also understand it a bit more. Understand the difference between our views. I think informal communication is a big key. Part of running a successful software, part of building it together.*" (P3).

"*A lot of the good ideas that we've had in the development has been because someone just happens to have a certain and great idea and goes ahead and*

> *spread the idea with someone. We'd have more possibilities with informal conversations with Ops as well; I think that would increase our ability to generate more ideas.*" (P4).

Another essential fact to keep an eye is to be clear and objective in the communication. Establish goals and objectives in which that communication has to reach. Express oneself (speak) clearly so that the other person can understand the information being sent. Be efficient, share context. P8 and P6 shared their concerns:

> "*I think that effective communication isn't about how frequently one does it or even what particular methods of communication are used, but it's about how effective is one's communication. If one is not an effective communicator then having lots of meetings is just a waste of time because one is just doing something that one is really bad at lots and lots of times. But if one is really good at it, meetings can be short, brief, and straight to the point, and one knows how to be effective with their time. I've never seen a team failed because they are too good at communication, but I've seen lots failed because they're not good enough.*" (P8).

> "*Success factor is a good contextualization. Often we are in our context, but we do not transfer that context to the other person before we begin to communicate. So, in my view, contextualization is a very important aspect, and it facilitates a lot. Another point is how the receiver of the message, how he can capture that information, because, in a communication that is not a conversation but on the chat, for example, you don't have intonation. Often, the way the recipient of the message understands that is different and may provoke some disagreement and discussion.*" (P6).

4.5.2 Major Challenges to Successful Communication

When asked about the major challenges for a successful communication in DevOps, 16 factors were mentioned by the participants (Figure 4.9 shows the complete list). The main challenges highlighted by the participants complement the list of factors for success. The most prominent factors were: geographic distance and organizational limitations; not enough technical knowledge, aggregated with not to understand each other's work; along with not empowering the teams, and not sharing enough information.

The most emphasized challenge were the silos, which could be formed by both geographic distance and/or organizational barriers. P5, P11, and P2 demonstrated how this often interferes with their communication:

> "*I think that's specifically location. So, if one is in a different location, it automatically creates a sort of silo. One can, of course, minimize the effect by having a chat with the operation team and stuff. But one is always going to have the operations team talking to themselves out of the channel and agreeing on stuff without involving the development team. So, I think presence is very important.*" (P5).

> "*I think distance, physical distance is a big problem. Because even if one is side by side, sometimes, one ends up missing information. So, if people are not close to each other, that already causes a big problem.*" (P11).

> "*So, when in one's company one has two silos, developers write the code and operations team applies it, developers detach from a reality of what it means to have a system running into production with maybe millions of database records. So, I think that basically, this is the silo separation between Devs and Ops that really leads to a lot of problems.*" (P2).

Another point that the participants considered is that people do not have enough technical skills to discuss issues with the other parties involved. Often Ops or Dev will start a conversation, but they do not have the technical knowledge to contribute to the same. This leads them to not worry about the other side of things and creates a culture of not empathizing with each other. P24 and P8 reported:

> "*So, one of them is a lack of domain knowledge in the development team. Building solutions for environments that developers don't really understand how value flows in that environment are very hard. Understanding why certain things work is important, understanding how the system get used, that is critical. If one doesn't*

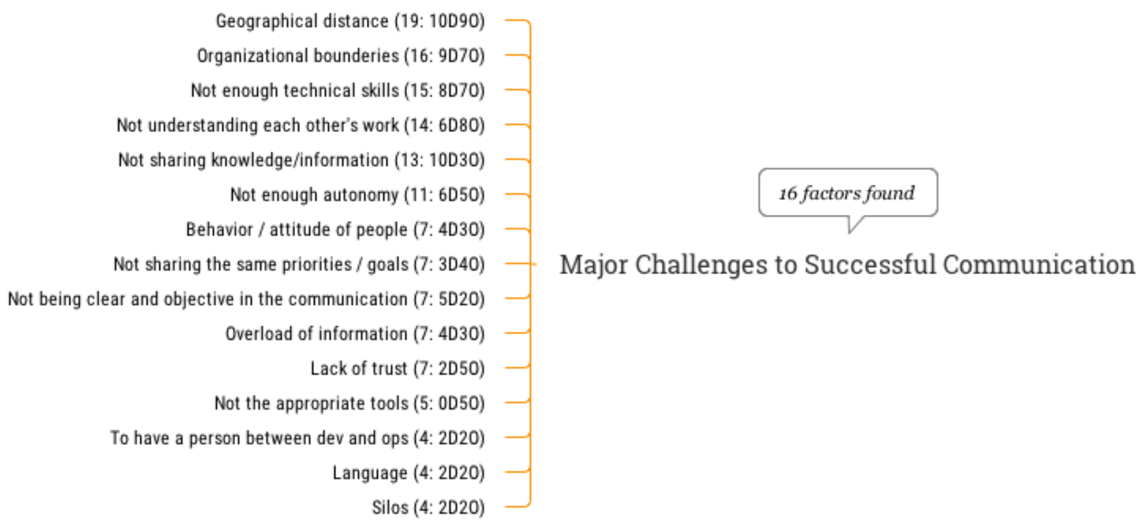

Geographical distance (19: 10D9O)
Organizational bounderies (16: 9D7O)
Not enough technical skills (15: 8D7O)
Not understanding each other's work (14: 6D8O)
Not sharing knowledge/information (13: 10D3O)
Not enough autonomy (11: 6D5O)
Behavior / attitude of people (7: 4D3O)
Not sharing the same priorities / goals (7: 3D4O)
Not being clear and objective in the communication (7: 5D2O)
Overload of information (7: 4D3O)
Lack of trust (7: 2D5O)
Not the appropriate tools (5: 0D5O)
To have a person between dev and ops (4: 2D2O)
Language (4: 2D2O)
Silos (4: 2D2O)
16 factors found
Major Challenges to Successful Communication

Figure 4.9 – Results: other findings - major challenges to successful communication

> *know how users are going to use the system, because one doesn't really understand their specific domain, it's going to be very hard to appreciate the challenges that they are going to have while using that system. And it doesn't matter which team one is, if one doesn't understand how to bring value to that system, it's going to be very difficult to build effectively operable software solutions.*" (P24).

> "*Assuming that these changes from the development team affect the operations. I think that the problem is that if one doesn't understand what one is running, then it's really hard to know how the impact of your changes, what impact your changes could have.*" (P8).

Not sharing the necessary information, leaving the teams in the dark, and not giving them the proper autonomy to perform their work also contribute to communication issues. P11 and P19 pointed out:

> "*In my view, another thing that I think is problematic, would be to be an information holder/hub, not having things documented, not sharing things.*" (P11).

> "*So we had a very heavy log structure that monitored the operations each team member did in terms of data, deploy, and code changes. So all those practices around that make it possible for people to have the information they need, when they need, regardless of role. Today, this does not work the same way, our developers do not have the same accesses as our infrastructure team has, and that is just because of an organizational definition.*" (P19).

Complementing the idea addressed by P19, two other participants also talked about the impact of organizational decisions on their routine and the way of communication by expressing that companies should be more careful with their organizational decisions and support. These can help boost the team's confidence to work even better. P16 and P11 elaborated:

> "*I think a problem is with the complexity of the communication process and also bureaucracy. Also, the lack of freedom in innovation in our environments, this just slow us down.*" (P16).

> "*My reality is that depending on the level of management, sometimes we have that institutional support we want, to be there, using the same new technologies as the market suggests. But in practice, when one has to solve a problem, sometimes one can't because one is missing that institutional support. So I think it makes a difference if one has an institutional support of the company, not only technical support.*" (P11).

5. DISCUSSION

This chapter discusses this study's findings. It also presents the main contributions of this study for research and for practice.

5.1 Discussion of Findings vs. Existing Literature

We have seen that communication is much more than just an exchange of information among interlocutors. Software engineering researchers have been investigating communication in diverse settings (e.g., global software engineering [87, 59], agile development [53, 1]). Our interest was on how communication takes place in DevOps. To recapitulate, our posed research question was as follows:

How does the two-way communication happen in a DevOps team?

To help answer the research question, we have searched in the literature a communication model / framework that focuses both on the personal aspects of the individuals involved in the communication process and on the aspects of communication itself. Within this search, we found the model presented by Mohr and Nevin [68]. In it, the authors introduce a series of factors to be observed in the communication, such as channel conditions (structure, climate, and power), communication strategies (frequency, direction, modality, and content), and channel outcomes (coordination, satisfaction, commitment, and performance). However, in our study, we sought to characterize communication in DevOps in light of their communication model. The channel conditions and communications strategies allowed us to tell how the communication process/practices take place, while the channel outcomes allowed us to tell about how each team member felt about those communication processes/practices.

5.1.1 What is a DevOps team?

Before diving into the discussion of the results mapped into the communication model, we first have to consider what a DevOps team is and if the way teams structure themselves impact in somehow their communication. Humble, back in 2012, published an online article [50] stating that "There's No Such Thing as a "Devops Team"." On that article, he explains why he thinks that there is no such thing by elaborating on the DevOps movement, developers and operations obligations towards their work, and how people are missing the concept of cross-functional team to use DevOps Team. However, by the end of his article, he says that if people want to call the teams that support the development – by building a platform that allows them to self-service environments for testing and production (and deployments to those environments) and provides metrics to the organization as a whole; by providing a toolchain that developers can use to build, test, deploy and run their systems; or by providing support and training for the platform and toolchain – a DevOps Team, he is fine with that.

That is Humble's point of view, and just like that, other authors [3, 22, 70, 90, 72, 10] have their own understanding of what a DevOps team is. Even though, we do not seek to define what is a DevOps team is or even if the same exists, we did ask our participants what a DevOps teams was for them and how their own teams were structured in order to have a better understanding of their context, shedding some light on the matter. Within their several examples of organization and team definitions, we group the results into three broad categories:

- Dev and Ops - same team: to have a cross-functional team (a team that has developers and operations personnel working together, located in the same team);
- DevOps Team - shared skill set: to a have a team of DevOps professionals (those who have both development and operations skills and can work on both ends);
- Dev Team and Ops Team - close collaboration: to have one development team and one operations team working together when necessary.

Regardless of their team's structure, company size, or nationality, participants reported very similar outcomes, the most relevant being: for communication to flow better, everyone must work together - avoid silos of people and knowledge -, seek more technical knowledge, and understand / empathize with the work of others. Most participants stating that cross-functional teams tend to work and communicate better because they have a better big picture of things.

One difference that we can observe regarding a cultural aspect of each participant is that Brazilian professionals tend to be more concerned with their work relationships than the Norwegians. While Brazilians cited more examples of group activities / interactions inside and outside of the work environment, Norwegians were more concerned about having efficient professionals that would be able to perform well when places in cross-functional teams. Nonetheless, all participants agreed that for a team to do its best, there must be a good level of interaction, stating that close relationship and networking comes naturally with their day to day work.

5.1.2 Mapping the Communication Process

Communication is an important and obvious way for team members to generate coordination processes in complex software development environments [32]. In order to fully commit to their tasks, people need to feel comfortable in their work environments and that only happens if they are well organized, meaning, have routines in place, trust their colleagues and feel empowered to do their job. To that end, Channel Conditions have a direct impact on how people coordinate their tasks and behave.

Therefore, having a proper way of structuring the communication becomes essential to the team's success. Participants have reported two ways of structuring it, a relational manner via formal meetings (mostly via Scrum or SAFE meetings) or a discrete manner via spontaneous meetings (dailies/stand-ups, *ad-hoc* meetings, pager-duty calls, etc.). On top of that, there are also other processes that organizations/teams are adopting to help structure their communication when geographical distance becomes a problem, such as warm handovers (to share more in-depth instructions of how to proceed moving forward), global meetings, and any other exchange program to facilitate the exchange of information.

Trust is one of the most important DevOps cultural enabler and the biggest challenge to be overcome. Trust is asymmetrical in that a single act of bad faith can destroy it, and repairing the damage, if possible at all, will take many small acts completed with competence and delivered as promised [22]. Even though participants reported nineteen factors that somehow contributed to the way they communicate, we believe that it is only after building trust that people begin to feel comfortable and are more willing to work together as a team, sharing the same goal, and helping each other.

Another aspect to consider is power. Empowering DevOps staff to understand that outcomes drive behavior raises the importance of assigning meaningful responsibilities [77]. Once people feel that they have responsibility for what they are working and that they are empowered with the information they need to do their job, they will think more carefully about the decision that are being made. Participants stated that power usually varies according to the situation, but almost all of them also reported that they would like to have more decision-power over things that impact their daily work, such as choose their own technology to work with, have the proper permissions/clearance to perform certain tasks, and to really have their voice heard and taken into account when deciding something that will directly affect them.

While knowing how to structure the communication and which aspects impact the most is essential, knowing when, where, and what to communicate is crucial. To have effective communication is vital to a smooth working process. Communication Strategies gather that information via the following communication facets: frequency (when), direction and modality (where), and content (what).

Frequency, or the amount of contact between channel members reflects how often channel members have contact with each other [69]. Participants reported several cases where the frequency of communication depended solely and exclusively on the content being communicated. If the matter was urgent, ad-hoc meetings were created, people who needed to be involved were immediately contacted in person (face-to-face) or even remotely. In some cases, depending on urgency, participants also mentioned the use of pagers to call attention and be more objective when needed. If the matter could wait to be communicated, then, it was done through already scheduled meetings. Another relevant point regarding frequency is that it has a direct impact on team's performance. Some participants have stated that communication with a very high frequency can cause a lot of noise and, due to that, information may be lost. On the other hand, if there is no good frequency of communication, people start to feel like working in the dark, which ends up generating frustrations.

Direction refers to the extent to which each party gives feedback or inputs to the other. It is important to note the direction in which the information is flowing to see if all parties involved are aware of information relevant to their tasks. In general, participants also said that the direction depends heavily on the content to be transmitted. Typically, the developers inform the operations that some service is down or not working properly and needs attention, while operations usually inform the developers of bugs found in the system or status of how the system is behaving after modifications introduced. One point that several participants brought to our attention is that communication, when it comes from management, is much more informative as to which tasks to be performed and about the decisions that were made in the project rather than asking for feedbacks or inputs on those, and that they would like to be able to share their opinions or concerns on those matters as well.

Modality refers to the means used to communicate, which can be informal (spontaneous) or formal (structured). Without questioning, the most favored means of communication is face-to-face, but as mentioned by Mishra, Mishra, and Ostrovska [67], while using only face-to-face communication, people could end up missing or not properly documenting relevant details exchanged in the communication. Thus, other media also reported were chats, formal meetings, e-mail and tickets. It is noteworthy that modality affects directly the teams' satisfaction and performance. While chats were the most cited due to their easy usability, features that can automate tasks, and quick feedback; e-mails and tickets were praised on their traceability, but were also criticized for the delay in return (the lack of more frequent updates).

Last but not least, content reflects what kind of information is being exchanged and can have a huge impact on the team's satisfaction and commitment towards their work. Indirect content such as exchange of technical information, discussions, and conversations not related to work were the most common and happen more frequently then direct communication. Direct content comes more often from management and refers to formal requests, conclusions, and obligations. An interesting fact pointed by some participants is that both sides (Devs and Ops) occasionally share recommendations with each other. Such recommendations being either how to better improve the performance via code or even what technologies would better support that application, which stimulates team work and increase satisfaction once they all feel responsible for the outcomes.

5.1.3 Recommendations for industry

DevOps aligns business requirements with IT performance, with the goal of adopting practices that allow a fast flow of changes to a production environment, while maintaining a high level of stability, reliability, and performance in these systems [40]. However, none of this would be possible without communication. Communication fosters collaboration which is one of the main principles behind DevOps. Observing the results found and review the literature, it is possible to find several studies ([12, 20, 40, 54, 71, 77]) that address the main challenges in the adoption of DevOps and suggest some actions to facilitate this transaction. While some of these actions also serve to assist the communication in DevOps, we have prepared a list with seven recommendations to improve communication in DevOps for the industry based on the main challenges listed by the participants. These recommendations are:

1. *Avoid silos:* To all costs, avoid organizational or personal silos. Encourage team members to work together, have them sharing the same goals. Team members working on a common goal should display mutual respect, grant assistance when needed, and develop other team members' ideas and contributions. A compelling goal can serve as a motivator for a cross-functional team working on a basic business issue. Practices: share the same goal, commit to the cause, efficiently coordinate the tasks.

2. *Build a trusting environment:* Every team member should have equal value, responsibilities, and be empowered to identify and solve their own problems and take decisions. If any team has assumed ownership of product they will have to learn to share that with everyone else on the team in order to succeed. When they have this sense of ownership, their behavior towards work is different, they become more empathetic towards each other – better understanding each other's work. Practices: pairing Dev and Ops, job rotation, pager-duty, social events.

3. *Share knowledge and technical skills:* Technical trust is build overtime of close collaboration. DevOps teams have to change their way of working and develop skills that will make possible mutual support towards Devs and Ops. Create process and procedures to allow team members to get to know each other technically, build a competence matrix and encourage them to share knowledge about their own specific com-

petencies. Practices: informal training, communities of practice, planned onboarding of new members, liftoffs, communication through Social Software (SoSo) to allow them to coordinate tasks and get to know each other informally as well.

4. *Get your communication structured:* Plan but also create room for some spontaneous (face-to-face) meetings. Create joint rooms (war rooms) for big features, releases, problem solving. Allow your team to do daily standups or weekly meetings to discuss technical issues. Promote tech talks and show cases to share knowledge and the team's achievements with other teams, collaboration is the key here.

5. *Tune the frequency of the communication:* Meetings are a necessary evil, plan accordingly but also ask for feedback from your team about their meetings effectiveness. Create open chats and let your teams discuss things in there at all times. Listen to the communication and act when needed.

6. *Promote awareness:* Allow your DevOps team to promote awareness on the content of the information flow in your communication. Get everyone to listen to the communication and express their concerns (everyone should be responsible for decisions). Let the team knows the importance of clear and objective communication. Different reports can come in handy to different people (ask for more emphasis on big launches/changes), still foster teamwork.

7. *Choose your tools wisely:* Find the best communication tools that work for your team. Later, make sure that everyone is onboard with its features and how to use them. Choose based on your needs but also watch out for some of the features, some background noise due to integration with other tools can be hindering.

5.2 Contributions for research

First of all, this study contributes directly to the SE area. We approached a topic (DevOps) that, even though it has been long camouflaged in industry, it is still considered new in academia. As stated in the methodology chapter (Chapter 3), most of the work that addresses DevOps has its origin in the industry, and none of them directly and exclusively addresses the process of communication.

Another interesting contribution of this work is that we used a theory to help construct the conceptual mapping on the subject. Also, it is worth highlighting that we have been able to use a communication theory focused on marketing within software engineering. The work related to communication within SE usually addresses communication challenges and by bringing the theory presented by Mohr and Nevin [68] into SE, it has allowed us to expand the aspects to be studied referring exclusively to communication.

Adding to the list of contributions, the fact that we have made presentations with partial results at large events has shown that there is space in both the industry and the academic research to address the issue in a scientific manner. The industry seems thrilled to be able to participate in research like ours by providing us evidence and feedback so that in the end we can also help them out with the analysis of the results found. Consequently, these presentations and feedback collected helped us in the validation and saturation of the content found.

Last but not least, to have a cross-culture study performed and obtaining similar results is a strong indication the research has a global contribution to the matter. Besides, even though we only reported 25 interviews (due to the saturation of information), there was still many people in the industry contacting us, who would like to contribute to the study, allowing for the possibility of extending this for future studies.

5.3 Contributions for practice

Thinking towards the industry, the main contribution of this work is to map how the communication takes place in teams that use DevOps. As mentioned in Chapter 2, in the Communication section, communication is much more than a just exchange of information between two interlocutors [25]. Therefore, the results here introduced to allow companies to observe the different aspects involved in a communication process.

Furthermore, this research contributes to industry by identifying challenges and solutions to improve communication in DevOps. In this aspect, it is worth mentioning that the examples cited by the participants facilitate the understanding of where the process may be failing, allowing companies to take actions right on the source of the problem. Also, those examples also help companies better comprehend the expectations set by their employees for a better working environment and the communication process to be used.

6. FINAL CONSIDERATIONS

This chapter elaborates a summary of the results found, revisiting the research question. Later, it presents the paper published about this work, its limitations and future work suggested.

6.1 Summary of Findings

This study presented the results from research that is trying to understand how the two-way communication happens in a DevOps team from the perceptions of different practitioners. This research went through two phases to accomplish its goal, one being a Literature Review on DevOps and Communication, the other being a Field Study where qualitative data were collected in the form of semi-structured interviews with 25 cross-culture DevOps practitioners. The results were mapped based on a communication model presented by Mohr and Nevin [68]. Table 6.1 summarizes the main results found.

Table 6.1: Summary of the findings

Aspect	**Findings**
	Channel Conditions
Structure	Team that are collocated and cross-functional tend to communicate better. The most used way of structuring the communication are via: face-to-face meetings, where frequency will depend on the duration/finance of the project; joint chat rooms, for talking about features, releases, status report, or just solving problems; daily stand-ups and weekly meetings, with availability for *ad-hoc* meetings as well; and Tech Talks / Show Cases, where everyone has the opportunity to gain knowledge on a technological aspect, or get on-board with new tasks being completed.
Climate	To keep teams engaged and communication flowing, it is important to create an environment in which they can trust each other and have their autonomy to work. With that established, it becomes easier to work together (helping and empathizing with each other) and sharing technical knowledge becomes natural.

continued . . .

. . . continued

Aspect	**Findings**
Power	Decision power and active voice will depend on the situation. However, it is essential to share this leadership and make decisions as a group, allowing everyone to be aware of all information relevant to the project.
	Communication Strategies
Frequency	If you are working, you are communicating. Being aware of all the information exchanged is crucial. The use of chat (individual or group) and daily meetings are the fastest way to keep informed. Meetings are a necessary evil. However, it is essential to be short and effective. Otherwise, they are just a waste of time for everyone involved.
Direction	The information exchange comes from all sides and also depends on the situation. However, it is necessary to ensure that the right people are getting the information they need to be able to work and that they also have the freedom to use the same means of communication to share feedback or collaborate in the matter.
Modality	Find the best communication tools that works for your team and your business. Make it easy for them to use and customize to their needs.
Content	Create awareness and contextualize all information, those are key points here. Everyone is interested in knowing about strategic plannings, technological advice, architectural changes, critical issues, etc.
	Other Findings
Factors of Success	Work together: share the same goal, share knowledge, empower people, commit to the work; Be empathetic towards each other: understand each other's work and try to get to know the person that you're working with; Be clear and objective in the communication.
Factors of Impediments	Silos (geographical or organizational) that prevents team from working together; Not having enough technical skills, therefore not understanding each other's work; Not empowering teams and not sharing enough information.

6.2 Publications

A paper submitted and accepted in early 2016. It is called "*Communication Challenges and Strategies in Distributed DevOps*" at International Conference of Global Software Engineering (ICGSE'16) [24].

6.3 Limitations

Engagement of industrial partners in research is a common challenge for the SE community. Our population is active DevOps practitioners. For this, a preliminary analysis of suggested participants was performed to ensure they were relevant. On an attempt to avoid selecting candidates that did not match our desired profile, an invitation letter and questionnaire were sent to the candidate participants for confirmation of relevance.

To minimize the concerns of the degree to which the results can be generalized to an overall population and setting, the target population was determined when designing the study, considering all context factors when drawing the conclusion about generalization. The participants selected represent different backgrounds, such as nationality, years of experience, current role, and company settings. Therefore, with the saturation of the results, findings can be generalized to that end.

A threat in interpreting the data is researcher bias. To avoid that bias, we decided to discuss and validate all interviews and analysis codes with other researchers and with the participants themselves. To minimize that threat, the preliminary results were presented in workshops and submitted to conferences.

Lastly, there is also a risk that our findings could be influenced by factors that escaped our attention. To mitigate this, we chose to have the evidence reviewed by more expert researchers and to perform agreement rounds in order to seek the completeness of the conclusions.

6.4 Future Work

The findings of this dissertation, its contributions, and limitations, indicate several possibilities for further research with relevance for SE theory, methodology and practice.

- Further research within the same companies: Further studies should be performed, where new data are collected from other DevOps practitioners that are actively working together (same company). Their opinions could be used to validate or further illustrate our conclusions presented in this dissertation.

- Further research collecting communication metrics: Further studies should be made to collect metrics on communication artifacts to validate our conclusions presented here. This would allow for a triangulation of the findings and a more accurate result.
- Further study of the recommendations: Further studies should be carried out to investigate how our recommendations are implemented in the companies, and whether the recommendations have had the intended effect.
- Further analysis of the communication model: With the future studies / research suggested above, it would be necessary to do a further analysis of the communication model used, studying the possibility of extending it with details focusing on the SE area.

There are certainly other directions for further research. However, the value of any such future work depends on the perceived relevance of the research problem of each particular audience. The results of the empirical research presented here support our claim that communication in DevOps teams has yet to be further investigated since several gaps were found with this exploratory research.

REFERENCES

[1] Alzoubi, Y. I.; Gill, A. Q. “Agile global software development communication challenges: A systematic review”. In: Proceedings of the Pacific Asia Conference on Information Systems, Chengdu, CN, AIS (Editor), 2014, pp. 1–13.

[2] Angelmar, R.; Stern, L. W. “Development of a content analytic system for analysis of bargaining communication in marketing”, *Journal of Marketing Research*, vol. 15–1, Feb 1978, pp. 93–102.

[3] Balalaie, A.; Heydarnoori, A.; Jamshidi, P. “Microservices architecture enables devops: migration to a cloud-native architecture”, *IEEE Software*, vol. 33–3, Mar 2016, pp. 42–52.

[4] Bang, S. K.; Chung, S.; Choh, Y.; Dupuis, M. “A grounded theory analysis of modern web applications: Knowledge, skills, and abilities for devops”. In: Proceedings of the Annual Conference on Research in Information Technology, Orlando, USA, ACM (Editor), 2013, pp. 61–62.

[5] Bennet Peter, D.; Association, A. M. “Dictionary of Marketing Terms”. Chicago, USA: NTC Business Books, 1995, 2 ed., vol. 1, 336p.

[6] Brazil, A. “Agile brazil”. Source: http://bit.ly/2iNCNDz, Sep 2016.

[7] Brown, A. D.; Starkey, K. “The effect of organizational culture on communication and information”, *Journal of Management Studies-Oxford*, vol. 31, Nov 1994, pp. 807–807.

[8] Bruneo, D.; Fritz, T.; Keidar-Barner, S.; Leitner, P.; Longo, F.; Marquezan, C.; Metzger, A.; Pohl, K.; Puliafito, A.; Raz, D.; Roth, A.; Salant, E.; Segall, I.; Villari, M.; Wolfsthal, Y.; Woods, C. “Cloudwave: Where adaptive cloud management meets devops”. In: Proceedings of the IEEE Symposium on Computers and Communication, Madeira, POR, IEEE (Editor), 2014, pp. 1–6.

[9] Buchbinder, E. “Beyond checking: Experiences of the validation interview”, *Qualitative Social Work*, vol. 10–1, Mar 2011, pp. 106–122.

[10] Calefato, F.; Lanubile, F. “A hub-and-spoke model for tool integration in distributed development”. In: Proceedings of the International Conference on Global Software Engineering, Orange County, USA, IEEE (Editor), 2016, pp. 129–133.

[11] Carmel, E.; Agarwal, R. “Tactical approaches for alleviating distance in global software development”, *IEEE Software*, vol. 18–2, Mar/Apr 2001, pp. 22–29.

[12] Colavita, F. “Devops movement of enterprise agile breakdown silos, create collaboration, increase quality, and application speed”. In: Proceedings of the International Conference in Software Engineering for Defence Applications, Rome, ITA, Ciancarini, P.; Sillitti, A.; Succi, G.; Messina, A. (Editors), 2016, pp. 203–213.

[13] ComputerWorld. “Computerworld foredragsholdere| without a collaborative culture, you don’t have devops”. Source: http://bit.ly/2iXj6En, Nov 2016.

[14] Corbin, J. “Basics of qualitative research : techniques and procedures for developing grounded theory”. Los Angeles, USA: Sage Publications, 2008, 3rd ed., 400p.

[15] Cripe, J. E.; Mansfield, S. R. “The value-added employee : 31 competencies to make yourself irresistible to any company”. Boston, USA: Butterworth-Heinemann, 2002, 2nd ed., 194p.

[16] Crowston, K.; Howison, J.; Masango, C.; Eseryel, U. Y. “The role of face-to-face meetings in technology-supported self-organizing distributed teams”, *IEEE Transactions on Professional Communication*, vol. 50–3, Aug 2007, pp. 185–203.

[17] Cruzes, D. S.; Dyba, T. “Recommended steps for thematic synthesis in software engineering”. In: Proceedings of the International Symposium on Empirical Software Engineering and Measurement, Banff, CAN, IEEE (Editor), 2011, pp. 275–284.

[18] Dainton, M.; Zelley, D. E. “Applying Communication Theory for Professional Life: A Practical Introduction”. California, USA: SAGE Publications, 2010, 2nd ed., 264p.

[19] Dance, F. E. X. “The "concept" of communication”, *Journal of Communication*, vol. 20–2, Jun 1970, pp. 201–210.

[20] de França, B. B. N.; Jeronimo Junior, H.; Travassos, G. H. “Characterizing devops by hearing multiple voices”. In: Proceedings of the Brazilian Symposium on Software Engineering, Maringa, BRA, ACM (Editor), 2016, pp. 53–62.

[21] Debois, P. “Devops: A software revolution in the making”, *Cutter IT Journal*, vol. 24–8, Aug 2011, pp. 3–5.

[22] DeGrandis, D. “Devops: So you say you want a revolution?”, *Cutter IT Journal*, vol. 24–8, Aug 2011, pp. 34.

[23] DevOpsDays. “Devopsdays oslo | communication in devops: A fairy tale or a horror story?” Source: http://bit.ly/2j7G67d, Sep 2016.

[24] Diel, E.; Marczak, S.; Cruzes, D. S. “Communication challenges and strategies in distributed devops”. In: Proceedings of the IEEE International Conference on Global Software Engineering, Orange County, USA, 2016, pp. 24–28.

[25] Duck, S.; McMahan, T. D. “The basics of communication : a relational perspective”. Thousand Oaks, USA: SAGE Publications, 2012, 2nd ed., 472p.

[26] Dwyer, F. R.; Walker Jr, O. C. “Bargaining in an asymmetrical power structure”, *The Journal of Marketing*, vol. 45, Winter 1981, pp. 104–115.

[27] Edwards, D. “Q&a: Lee thompson, former chief technologist of e*trade financial | dev2ops”. Source: http://bit.ly/1Q37Uno, Nov 2016.

[28] Edwards, D. “What is devops? | dev2ops”. Source: http://bit.ly/1pB9cF8, Sep 2016.

[29] Elliott, J. J. “Design of a product-focused customer-oriented process”, *Information and Software Technology*, vol. 42–14, Nov 2000, pp. 973–981.

[30] Erich, F.; Amrit, C.; Daneva, M. “Cooperation between information system development and operations: a literature review”. In: Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, Torino, ITA, ACM (Editor), 2014, pp. 69.

[31] Erich, F.; Amrit, C.; Daneva, M. “A mapping study on cooperation between information system development and operations”. In: Proceedings of the International Conference on Product-Focused Software Process Improvement, Helsinki, FIN, Jedlitschka, A.; Kuvaja, P.; Kuhrmann, M.; Männistö, T.; Münch, J.; Raatikainen, M. (Editors), 2014, pp. 277–280.

[32] Espinosa, J. A.; Carmel, E. “The impact of time separation on coordination in global software teams: a conceptual foundation”, *Software Process: Improvement and Practice*, vol. 8–4, Sep 2003, pp. 249–266.

[33] Etgar, M. “Effects of administrative control on efficiency of vertical marketing systems”, *Journal of Marketing Research*, vol. 13–1, Feb 1976, pp. 12–24.

[34] Falcione, L. R.; Sussman, L.; Herden, R. “Communication climate in organizations”. In: *Handbook of Organizational Communication: An Interdisciplinary Perspective*, Jablin, M. F.; Putnam, L. L.; Roberts, K.; Porter, L. (Editors), Newbury Park, CA: Sage Publications, 1987, chap. 7, pp. 195–227.

[35] Farace, R. V.; Monge, P. R.; Russell, H. H. “Comunicating and organizing”. Massachusetts, USA: Addison-Wesley, 1977, 1 ed., 281p.

[36] Fitzgerald, B.; Stol, K.-J. “Continuous software engineering: A roadmap and agenda”, *Journal of Systems and Software*, vol. 123, Jul 2015, pp. 176–189.

[37] Fitzpatrick, L.; Dillon, B.; Dillon, M.; Dillon, S.; Dillon, T. “The business case for devops: A five-year retrospective”, *Cutter IT Journal*, vol. 24–8, Aug 2011, pp. 19.

[38] Frazier, G. L. “Interorganizational exchange behavior in marketing channels: a broadened perspective”, *The Journal of Marketing*, vol. 47–4, Autumn 1983, pp. 68–78.

[39] Goles, T.; Chin, W. W. “Information systems outsourcing relationship factors: detailed conceptualization and initial evidence”, *ACM Special Interest Group on Management Information Systems*, vol. 36–4, Fall 2005, pp. 47–67.

[40] Gottesheim, W. “Challenges, benefits and best practices of performance focused devops”. In: Proceedings of the International Workshop on Large-Scale Testing, Austin, USA, ACM (Editor), 2015, pp. 3–3.

[41] Guetzkow, H. S. “Communications in organizations”. Indianopolis, USA: Bobbs-Merrill, 1967, 1 ed., 534-573p.

[42] Haight, C. “Devops: Born in the cloud and coming to the enterprise”, *Gartner Research, ID*, –G00208163, Oct 2010.

[43] Hamunen, J. “Challenges in adopting a devops approach to software development and operations”, Master thesis, Aalto University, 2016, 69p.

[44] Henttonen, K.; Blomqvist, K. “Managing distance in a global virtual team: the evolution of trust through technology-mediated relational communication”, *Strategic Change*, vol. 14–2, Apr 2005, pp. 107–119.

[45] Herbsleb, J. D.; Grinter, R. E. “Splitting the organization and integrating the code: Conway’s law revisited”. In: Proceedings of the International Conference on Software Engineering, Los Angeles, USA, ACM (Editor), 1999, pp. 85–95.

[46] Herbsleb, J. D.; Mockus, A. “An empirical study of speed and communication in globally distributed software development”, *IEEE Transactions on Software Engineering*, vol. 29–6, Jun 2003, pp. 481–494.

[47] Holmstrom, H.; Conchuir, E. O.; Agerfalk, P. J.; Fitzgerald, B. “Global software development challenges: A case study on temporal, geographical and socio-cultural distance”. In: Proceedings of the IEEE International Conference on Global Software Engineering, Florianopolis, BRA, IEEE (Editor), 2006, pp. 3–11.

[48] Hosono, S.; He, J.; Liu, X.; Li, L.; Huang, H.; Yoshino, S. “Fast development platforms and methods for cloud applications”. In: Proceedings of the Asia-Pacific Services Computing Conference, Jeju, KOR, IEEE (Editor), 2011, pp. 94–101.

[49] Hosono, S.; Shimomura, Y. “Application lifecycle kit for mass customization on paas platforms”. In: Proceedings of the World Congress on Services, Honolulu, USA, 2012, pp. 397–398.

[50] Humble, J. “There’s no such thing as a "devops team"”. Source: http://bit.ly/1VXFZIs, Jun 2016.

[51] Humble, J.; Farley, D. “Continuous delivery: reliable software releases through build, test, and deployment automation”. Boston, USA: Pearson Education, 2010, 1 ed., 512p.

[52] Humble, J.; Molesky, J. “Why enterprises must adopt devops to enable continuous delivery”, *Cutter IT Journal*, vol. 24–8, Aug 2011, pp. 6.

[53] Hummel, M.; Rosenkranz, C.; Holten, R. “The role of communication in agile systems development”, *Business & Information Systems Engineering*, vol. 5–5, Jul 2013, pp. 343–355.

[54] Hussaini, S. W. “Strengthening harmonization of development (dev) and operations (ops) silos in it environment through systems approach”. In: Proceedings of the International IEEE Conference on Intelligent Transportation Systems, Barcelona, SPA, IEEE (Editor), 2014, pp. 178–183.

[55] Hüttermann, M. “Quality and testing”. In: *DevOps for Developers*, Berkeley, CA: Apress, 2012, vol. 1, chap. 4, pp. 51–64.

[56] ICGSE. “Icgse 2016 program| communication challenges and strategies in distributed devops”. Source: http://bit.ly/2jZVqBd, Jul 2016.

[57] Jabbari, R.; bin Ali, N.; Petersen, K.; Tanveer, B. “What is devops?: A systematic mapping study on definitions and practices”. In: Proceedings of the Scientific Workshop, Edinburgh, SCO, ACM (Editor), 2016, pp. 12:1–12:11.

[58] Keyworth, B. “Where is it operations within devops?”, *Cutter IT Journal*, vol. 24–12, Jan 2011, pp. 12.

[59] Khan, A.; Basri, S.; Amin, F.; Teknologi, U.; Perak, T.; Studies, I. “Communication risks and best practices in global software development during requirements change management: A systematic literature review protocol”, *Research Journal of Applied Sciences, Engineering and Technology*, vol. 6, Oct 2013, pp. 3514–3519.

[60] Khan, A. A.; Basri, S.; Dominic, P. D. D. “Communication risks in gsd during rcm: Results from slr”. In: Proceedings of the International Conference on Computer and Information Sciences, Kuala Lumpur, MAL, IEEE (Editor), 2014, pp. 1–6.

[61] Kitchenham, B. A.; Pfleeger, S. L. “Personal opinion surveys”. In: *Guide to Advanced Empirical Software Engineering*, Shull, F.; Singer, J.; Sjøberg, D. I. K. (Editors), London, UK: Springer, 2008, chap. 3, pp. 63–92.

[62] Kraut, R. E.; Streeter, L. A. “Coordination in software development”, *Communications of the ACM*, vol. 38–3, Mar 1995, pp. 69–81.

[63] Krusche, S.; Alperowitz, L. “Introduction of continuous delivery in multi-customer project courses”. In: Proceedings of the International Conference on Software Engineering, Hyderabad, IND, ACM (Editor), 2014, pp. 335–343.

[64] Lengel, R.; Daft, R. “The relationship between message content and media selection in managerial communication: Some preliminary evidence”, Technical report, Texas University, 1985, 355-366p.

[65] Macneil, I. R. “Economic analysis of contractual relations: its shortfalls and the need for a rich classificatory apparatus”, *Northwestern University Law Review*, vol. 75, 1980, pp. 1018.

[66] Miles, B. M.; Huberman, M. A. “Qualitative data analysis : an expanded sourcebook”. Thousand Oaks, USA: Sage Publications, 1994, 2nd ed., 352p.

[67] Mishra, D.; Mishra, A.; Ostrovska, S. “Impact of physical ambiance on communication, collaboration and coordination in agile software development: An empirical evaluation”, *Information and Software Technology*, vol. 54–10, Oct 2012, pp. 1067–1078.

[68] Mohr, J.; Nevin, J. R. “Communication strategies in marketing channels: A theoretical perspective”, *The Journal of Marketing*, vol. 54–4, Oct 1990, pp. 36–51.

[69] Mohr, J. J.; Sohi, R. S. “Communication flows in distribution channels: Impact on assessments of communication quality and satisfaction”, *Journal of Retailing*, vol. 71–4, Nov 1996, pp. 393–415.

[70] Mueller, E. “Devops and the people who practice it: Winning their hearts and minds”, *Cutter IT Journal*, vol. 24–12, Dec 2011, pp. 6.

[71] Nagpal, S.; Shadab, A. “Literature review: Promises and challenges of devops”, Technical report, University of Waterloo, 2017, 8p.

[72] Nybom, K.; Smeds, J.; Porres, I. “On the impact of mixing responsibilities between devs and ops”. In: Proceedings of the International Conference on Agile Software Development, Edinburgh, SCO, Springer (Editor), 2016, pp. 131–143.

[73] Olson, G. M.; Olson, J. S. “Distance matters”, *Human-Computer Interaction*, vol. 15–2, Sep 2000, pp. 139–178.

[74] Paasivaara, M.; Lassenius, C. “Collaboration practices in global inter-organizational software development projects”, *Software Process: Improvement and Practice*, vol. 8–4, Sep 2003, pp. 183–199.

[75] Perez, J. F.; Wang, W.; Casale, G. “Towards a devops approach for software quality engineering”. In: Proceedings of the Workshop on Challenges in Performance Methods for Software Development, Austin, USA, ACM (Editor), 2015, pp. 5–10.

[76] Phifer, B. “Next-generation process integration: Cmmi and itil do devops”, *Cutter IT Journal*, vol. 24–8, Aug 2011, pp. 28.

[77] Rajkumar, M.; Pole, A. K.; Adige, V. S.; Mahanta, P. “Devops culture and its impact on cloud delivery and software development”. In: Proceedings of the International Conference on Advances in Computing, Communication, & Automation, Dehradun, IND, IEEE (Editor), 2016, pp. 1–6.

[78] Sacks, M. “Pro Website Development and Operations: Streamlining DevOps for Large-scale Websites”. Berkely, USA: Apress, 2012, 1st ed., 124p.

[79] Schmidt, K.; Simonee, C. “Coordination mechanisms: Towards a conceptual foundation of cscw systems design”, *Computer Supported Cooperative Work*, vol. 5–2, Jun 1996, pp. 155–200.

[80] Seaman, B. C. “Qualitative methods in empirical studies of software engineering”. In: *Guide to Advanced Empirical Software Engineering*, Shull, F.; Singer, J.; Sjøberg, I. K. D. (Editors), New York, USA: Springer, 2008, chap. 2, pp. 35–62.

[81] Shafer, A. “Agile infrastructure velocity 09”. Source: http://bit.ly/1KbKkP6, Nov 2016.

[82] Shang, W. “Bridging the divide between software developers and operators using logs”. In: Proceedings of the International Conference on Software Engineering, Porto Alegre, BRA, IEEE (Editor), 2012, pp. 1583–1586.

[83] Shang, W.; Jiang, Z. M.; Adams, B.; Hassan, A. E.; Godfrey, M. W.; Nasser, M.; Flora, P. “An exploratory study of the evolution of communicated information about the execution of large software systems”, *Journal of Software: Evolution and Process*, vol. 26–1, Nov 2014, pp. 3–26.

[84] Sill, A. “Cloud standards and the spectrum of development”, *IEEE Cloud Computing*, –3, Sep 2014, pp. 15–19.

[85] Singer, J.; Sim, E. S.; Lethbridge, C. T. “Software engineering data collection for field studies”. In: *Guide to advanced empirical software engineering*, Shull, F.; Singer, J.; Sjøberg, I. K. D. (Editors), New York, USA: Springer, 2008, chap. 1, pp. 9–34.

[86] Smeds, J.; Nybom, K.; Porres, I. “Devops: a definition and perceived adoption impediments”. In: Proceedings of the International Conference on Agile Software Development, Edinburgh, SCO, Springer (Editor), 2015, pp. 166–177.

[87] Smite, D. “Global software development projects in one of the biggest companies in latvia: is geographical distribution a problem?”, *Software Process: Improvement and Practice*, vol. 11–1, Mar 2006, pp. 61–76.

[88] Swartout, P. “Continuous Delivery and DevOps–A Quickstart Guide”. Birmingham, UK: Packt Publishing Ltd, 2014, 12th ed., 154p.

[89] Syed, M. H.; Fernandez, E. B. “Cloud ecosystems support for internet of things and devops using patterns”. In: Proceedings of the IEEE International Conference on Internet-of-Things, Design, and Implementation, Berlin, GER, IEEE (Editor), 2016, pp. 301–304.

[90] Tamburri, D. A.; Kazman, R.; Fahimi, H. “The architect’s role in community shepherding”, *IEEE Software*, vol. 33–6, Oct 2016, pp. 70–79.

[91] Tarone, E. “Some thoughts on the notion of communication strategy*”, *TESOL Quarterly*, vol. 15–3, Sep 1981, pp. 285–295.

[92] Ulrich, D. “Tie the corporate knot: Gaining complete customer commitment”, *MIT Sloan Management Review*, vol. 30–4, Summer 1989, pp. 19.

[93] Wahaballa, A.; Wahballa, O.; Abdellatief, M.; Xiong, H.; Qin, Z. “Toward unified devops model”. In: Proceedings of the IEEE International Conference on Software Engineering and Service Science, Beijing, CHI, IEEE (Editor), 2015, pp. 211–214.

[94] Wettinger, J.; Breitenbücher, U.; Leymann, F. “Devopslang–bridging the gap between development and operations”. In: *Service-Oriented and Cloud Computing*, Villari, M.; Zimmermann, W.; Lau, K.-K. (Editors), Springer, 2014, *LNCS*, vol. 8745, chap. 8, pp. 108–122.

[95] Wettinger, J.; Breitenbücher, U.; Leymann, F. “Standards-based devops automation and integration using tosca”. In: Proceedings of the IEEE/ACM International Conference on Utility and Cloud Computing, London, UK, IEEE (Editor), 2014, pp. 59–68.

[96] Winkler, J. K.; Dibbern, J.; Heinzl, A. “The impact of cultural differences in offshore outsourcing—case study results from german–indian application development projects”, *Information Systems Frontiers*, vol. 10–2, Feb 2008, pp. 243–258.

[97] Wohlin, C. “Guidelines for snowballing in systematic literature studies and a replication in software engineering”. In: Proceedings of the International Conference on Evaluation and Assessment in Software Engineering, London, UK, ACM (Editor), 2014, pp. 38.

[98] Womack, J. P.; Jones, D. T. “Lean thinking: banish waste and create wealth in your corporation”. New York, USA: Simon and Schuster, 2010, 2nd ed., 396p.

[99] Wood, J. “Communication Theories in Action: An Introduction”. Massachusetts, USA: Wadsworth, 2004, 3rd ed., 400p.

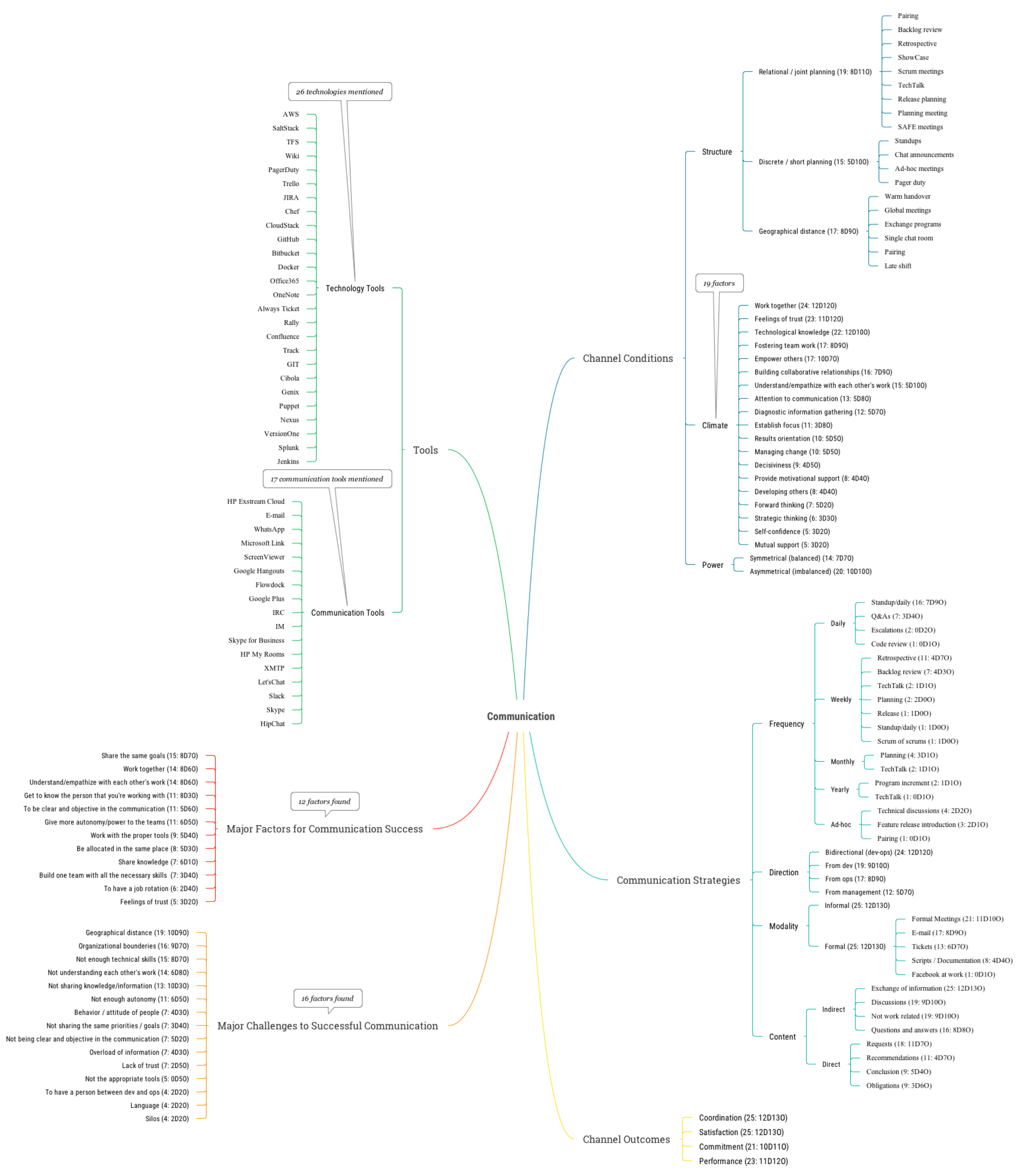
[100] Yin, R. “Case study research : design and methods”. Los Angeles, USA: SAGE Publications, 2013, 5th ed., 312p.

APPENDIX A – SCRIPT INTERVIEW

Interview:

1. What is DevOps for you?
2. Can you describe how communication between Dev and Ops teams usually happens within your company?
3. What are the roles that participate in this exchange of information? How does it happen (for email, face-to-face chat, etc.)? What tools are used?
4. What kind of information do you usually receive / send to / from the other team? Could you provide some examples please? How does this affect the daily activities of the team?
5. How often does the communication happen? How long does it take?
6. Would you have an example where the frequency of communication had an impact (positive or negative) in the performance of your team?
7. Do you know if there is any meeting where both teams are invited to participate (product release, etc.)? What is usually shared in these meetings? Are they productive? How?
8. In your opinion, do you think that the communication between both teams flows well? Do you understand that there is a trust between these teams? How it impacts the work of your team?
9. Do you remember some event where communication between the teams was compromised? Could you share it? (E.g.: time change, tooling, new team members, etc.)
10. In your opinion, do you think that Devs communicate more to Ops or otherwise? How does this impact your team?
11. Which team has the most active voice? (Devs vs. Ops) Can you tell us a situation where it has become clear? How does this impact your team?
12. Do you have other opportunities within the team and the company to talk to Dev and Ops (for example, through chat rooms)? Would you have any examples of when this type of communication was most needed?
13. Would you like to have more opportunities for informal conversations between Devs and Ops? Which opportunities you would like to have at hand? How do you think this could happen and what effect this might have on their team activities?
14. Do you think the fact of being part of the same organization (or not) changes the way you communicate with Dev / Ops? Would you have any examples of how it impacts the work of your team?
15. In your opinion, does it make a difference to the daily activities of the team the fact that Dev and Ops are or not physically in the same place? Do you think this changes anything in the communication process?
16. What do you think are the major impediments for good communication between Devs and Ops?
17. What do you think are the major factor for success in communication between Devs and Ops?

APPENDIX B – MIND MAP OF THE RESULTS


Communication
Tools
Technology Tools
26 technologies mentioned
AWS
SaltStack
TFS
Wiki
PagerDuty
Trello
JIRA
Chef
CloudStack
GitHub
Bitbucket
Docker
Office365
OneNote
Always Ticket
Rally
Confluence
Track
GIT
Cibola
Genix
Puppet
Nexus
VersionOne
Splunk
Jenkins
Communication Tools
17 communication tools mentioned
HP Exstream Cloud
E-mail
WhatsApp
Microsoft Link
ScreenViewer
Google Hangouts
Flowdock
Google Plus
IRC
IM
Skype for Business
HP My Rooms
XMTP
Let'sChat
Slack
Skype
HipChat
Major Factors for Communication Success
12 factors found
Share the same goals (15: 8D7O)
Work together (14: 8D6O)
Understand/empathize with each other's work (14: 8D6O)
Get to know the person that you're working with (11: 8D3O)
To be clear and objective in the communication (11: 5D6O)
Give more autonomy/power to the teams (11: 6D5O)
Work with the proper tools (9: 5D4O)
Be allocated in the same place (8: 5D3O)
Share knowledge (7: 6D1O)
Build one team with all the necessary skills (7: 3D4O)
To have a job rotation (6: 2D4O)
Feelings of trust (5: 3D2O)
Major Challenges to Successful Communication
16 factors found
Geographical distance (19: 10D9O)
Organizational bounderies (16: 9D7O)
Not enough technical skills (15: 8D7O)
Not understanding each other's work (14: 6D8O)
Not sharing knowledge/information (13: 10D3O)
Not enough autonomy (11: 6D5O)
Behavior / attitude of people (7: 4D3O)
Not sharing the same priorities / goals (7: 3D4O)
Not being clear and objective in the communication (7: 5D2O)
Overload of information (7: 4D3O)
Lack of trust (7: 2D5O)
Not the appropriate tools (5: 0D5O)
To have a person between dev and ops (4: 2D2O)
Language (4: 2D2O)
Silos (4: 2D2O)
Channel Conditions
Structure
Relational / joint planning (19: 8D11O)
Pairing
Backlog review
Retrospective
ShowCase
Scrum meetings
TechTalk
Release planning
Planning meeting
SAFE meetings
Discrete / short planning (15: 5D10O)
Standups
Chat announcements
Ad-hoc meetings
Pager duty
Geographical distance (17: 8D9O)
Warm handover
Global meetings
Exchange programs
Single chat room
Pairing
Late shift
Climate
19 factors
Work together (24: 12D12O)
Feelings of trust (23: 11D12O)
Technological knowledge (22: 12D10O)
Fostering team work (17: 8D9O)
Empower others (17: 10D7O)
Building collaborative relationships (16: 7D9O)
Understand/empathize with each other's work (15: 5D10O)
Attention to communication (13: 5D8O)
Diagnostic information gathering (12: 5D7O)
Establish focus (11: 3D8O)
Results orientation (10: 5D5O)
Managing change (10: 5D5O)
Decisiviness (9: 4D5O)
Provide motivational support (8: 4D4O)
Developing others (8: 4D4O)
Forward thinking (7: 5D2O)
Strategic thinking (6: 3D3O)
Self-confidence (5: 3D2O)
Mutual support (5: 3D2O)
Power
Symmetrical (balanced) (14: 7D7O)
Asymmetrical (imbalanced) (20: 10D10O)
Communication Strategies
Frequency
Daily
Standup/daily (16: 7D9O)
Q&As (7: 3D4O)
Escalations (2: 0D2O)
Code review (1: 0D1O)
Weekly
Retrospective (11: 4D7O)
Backlog review (7: 4D3O)
TechTalk (2: 1D1O)
Planning (2: 2D0O)
Release (1: 1D0O)
Standup/daily (1: 1D0O)
Scrum of scrums (1: 1D0O)
Monthly
Planning (4: 3D1O)
TechTalk (2: 1D1O)
Yearly
Program increment (2: 1D1O)
TechTalk (1: 0D1O)
Ad-hoc
Technical discussions (4: 2D2O)
Feature release introduction (3: 2D1O)
Pairing (1: 0D1O)
Direction
Bidirectional (dev-ops) (24: 12D12O)
From dev (19: 9D10O)
From ops (17: 8D9O)
From management (12: 5D7O)
Modality
Informal (25: 12D13O)
Formal (25: 12D13O)
Formal Meetings (21: 11D10O)
E-mail (17: 8D9O)
Tickets (13: 6D7O)
Scripts / Documentation (8: 4D4O)
Facebook at work (1: 0D1O)
Content
Indirect
Exchange of information (25: 12D13O)
Discussions (19: 9D10O)
Not work related (19: 9D10O)
Questions and answers (16: 8D8O)
Direct
Requests (18: 11D7O)
Recommendations (11: 4D7O)
Conclusion (9: 5D4O)
Obligations (9: 3D6O)
Channel Outcomes
Coordination (25: 12D13O)
Satisfaction (25: 12D13O)
Commitment (21: 10D11O)
Performance (23: 11D12O)